Fachhochschule Düsseldorf

University of Applied Sciences

Fachbereich 3: Elektrotechnik

Vertiefungsrichtung: Automatisierungstechnik

Wintersemester 2011/2012

Düsseldorf, den 05.03.2012

PRAXISPROJEKT

Entwicklung eines amperometrischen Blutglukosemessgerätes mit Touchscreen und integriertem Funkmodul zur Fernkontrolle für AAL

Erstellt von:

Tanja Kleiner
Matrikelnummer: 530197

Betreuer an der FH- Düsseldorf:

Prof. Dr.-Ing. Ulrich G. Schaarschmidt
Prof. Dr. rer. nat. Wolfgang Lux

Inhaltsverzeichnis

1 Vorwort

"Ja, lang leben will halt alles, aber alt werden will kein Mensch." [1], bemerkte Johann Nepomuk Nestroy in seinem Werk „Die Anverwandten“, welches er 1848 veröffentlichte, und auch heute noch scheint diese Feststellung des österreichischen Schauspielers, Satirikers, Dramatikers und Sängers einen wahren Kern zu haben.

Seit vielen Jahren schon fehlt in Deutschland eine große Anzahl von Pflegekräften. Natürlich hat dies zur Folge, dass der Unmut der pflegebedürftigen Personen wächst, da das Pflegepersonal sich nicht in dem Umfang um ihre Patienten kümmern kann, wie es eigentlich nötig wäre. Abhilfe schafft hier ein Projekt namens „AAL“ („Ambient Assisted Living“), wodurch es älteren Mitmenschen gestattet ist, in ihrem gewohnten Umfeld zu leben und trotzdem medizinisch betreut zu werden. Nicht durch Pflegekräfte, welche tagtäglich nach dem Rechten sehen, auch nicht durch tägliche Arztbesuche, sondern durch die fortschrittlichen Möglichkeiten, welche die Telemedizin bietet. Dieses Fachgebiet der Telematik befasst sich mit „Diagnostik und Therapie unter Überbrückung einer räumlichen oder auch zeitlichen („asynchron“) Distanz zwischen Arzt (Telearzt), Apotheker und Patienten oder zwischen zwei sich konsultierenden Ärzten mittels Telekommunikation.“ [2].

Je nachdem wie groß der Betreuungsbedarf des Patienten ist, kann diese Art der Pflege individuell angepasst werden. Manch einer braucht lediglich eine Grundsicherheit: zum Beispiel einen Notfallknopf, welcher im Gefahrfall Hilfe rufen kann, einen Rauchmelder, Geräte zur Messung von beispielsweise Blutdruck oder Blutglukose, oder sonstige ‚kleiner Helfer für den Alltag‘. Mit genau solchen Technologien befasst sich das AAL- Projekt der Fachhochschule Düsseldorf. Bereits existierende Geräte werden analysiert und nachgebaut. Dies ist allerdings noch nicht alles: Um sie an die Notwendigkeiten, welche das AAL-Projekt voraussetzt anzupassen, werden diese Gerätschaften mit einem Funkmodul ausgestattet. Dieses erlaubt es, die Daten an einen PC zu senden, von dem aus dann weitere Schritte eingeleitet werden können- Man kann die Daten an den behandelnden Arzt übermitteln lassen, der- zum Beispiel bei der Messung des Blutzuckers- nachverfolgen kann, wie sich die Daten der Messungen verändern, sie speichern und nachträglich erneut ansehen. So wird nicht nur den älteren Mitmenschen das Leben erleichtert, da die Anzahl der Arztbesuche minimiert wird. Der Kostenfaktor spielt hierbei auch eine große Rolle. Denn die Zeit, die der Arzt ansonsten damit verbringen würde, die Messungen am Patienten selbst durchzuführen, kann er nun anders einteilen.

Anforderungen an die Entwickler von AAL-Komponenten sind hier unter anderem die Erstellung einer leicht verständlichen Benutzeroberfläche, welche an die Zielgruppe angepasst ist. Man darf nicht vergessen, dass einige ältere Mitbürger im Umgang mit dem PC wenig bis gar nicht routiniert sind, außerdem müssen etwaige Seh- oder Hörstörungen bedacht werden.

Problematisch bei der Erforschung neuer Technologien ist hierbei zum einen, dass es keine international anerkannten Standards gibt. Somit sind Geräte verschiedener Hersteller unter Umständen nicht miteinander kombinierbar bzw. austauschbar. Um dieses Problem etwas einzugrenzen, wird häufig auf die offene Gateway-Plattform OSGi als Ansatz zurückgegriffen. Zum anderen gibt es auch beim Thema Datenschutz noch einige ungeklärte Fragen: Wie sollen die Patientendaten gesichert an den Arzt zugestellt werden? Welche Daten dürfen überhaupt übertragen werden? Und nicht zuletzt: Kann der Patient jederzeit einsehen, was gesendet wird und dies mitbestimmen?

Diese Arbeit befasst sich nun im Hinblick auf das AAL-Projekt mit der Entwicklung eines Blutglukosemessgerätes mit Funkmodul zur Übertragung der Messdaten an den PC des Patienten, bzw. des Betreuers. Es soll ein Einblick in die verschiedenen Arten der Blutzuckermessung gegeben werden. Der Leser soll ausserdem an den Weg von der Analyse des Messverfahrens bis hin zum fertigen Messgerät herangeführt werden und die einzelnen Schaltungsteile, sowie die Schaltung und deren Funktionsart im Ganzen kennenlernen.

2 Blutglukose – Medizinische Grundlagen

Allgemein betrachtet versteht man unter Blutzucker den Glukoseanteil im Blut (Blutglukose). Dabei ist Glukose vor allem für das Gehirn wichtig, doch auch Fett- oder Muskelzellen greifen auf diesen Energielieferanten zurück. Glukose wird hauptsächlich mit der Nahrung über den Darm ins Blut transportiert, sie kann aber auch aus anderen Kohlenhydraten durch Um- oder Abbau durch den Körper gewonnen werden. Damit die Blutglukose von den Zellen aufgenommen und weiterverarbeitet werden kann, wird von der Bauchspeicheldrüse das Hormon *Insulin* produziert. Ohne Insulin ist das Gelangen der Glukose in die Zellen nicht möglich und somit steigt der Blutzuckerspiegel an. In Abbildung 1 ist ein Beispiel für den Verlauf von Blutglukose und Insulin im Tagesverlauf dargestellt.

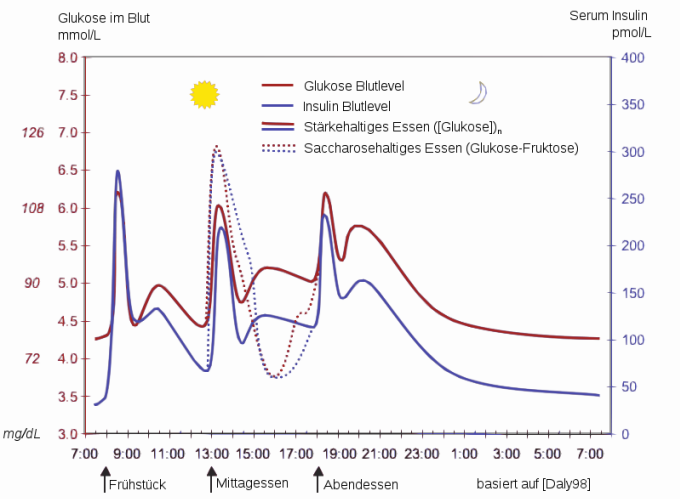


Abbildung 1: Schwankungen von Blutzucker (rot) und Insulin (blau) im Tagesverlauf

Der Blutzuckerwert ist ein wichtiger Messwert. Ist dieser dauerhaft zu hoch, spricht man von *Hyperglykämie*, bei einem zu niedrigen Wert von *Hypoglykämie*. Die Normalwerte beim Menschen betragen nüchtern etwa 70-99mg/dl* (entsprechen 3,9-5,5 mmol/l*), nach einer kohlenhydratreichen Nahrungsaufnahme circa 160 mg/dl (entsprechen 8,9 mmol/l) und 140mg/dl (7,8 mmol/l) zwei Stunden nach der Mahlzeit (vgl. [10]).

Hyperglykämie, also ein zu hoher Blutglukosespiegel kann eintreten bei Diabetes mellitus, einer Erkrankung der Bauchspeicheldrüse, beispielsweise bei Krebs („Pankreaskarzinom“) oder bei chronischen Entzündungen („Pankreatitis“), Hormondrüsenkrankheiten, wie zum Beispiel einer zu hohen Menge an Kortisol im Blut („Hyperkortisolismus“), durch bestimmte Medikamente oder auch bei Schwangeren. Symptome für einen zu hohen Blutzuckerspiegel sind vermehrter Harndrang, sowie starkes Durstgefühl, ein trockener Mund und trockene Haut, außerdem Übelkeit, Schwächeanfälle, Verwirrtheit, Sehstörungen, Schwindel und in sehr ausgeprägten Fällen sogar ein hyperglykämisches Koma.

Hypoglykämie kann durch Überdosierung von Insulin oder anderen Medikamenten, die man zur

*) Der Blutglukosewert wird entweder in Milligramm pro Deziliter (mg/dl) oder in Millimol pro Liter (mmol/l) angegeben. Beide Formen sind wissenschaftlich anerkannt und werden von modernen Messgeräten verwendet. Der Umrechnungsfaktor beträgt:
1 mmol/l = 18,02 mg/dl oder 1 mg/dl = 1/18,02 mmol/l

Bekämpfung von Diabetes nimmt, entstehen. Es kann aber auch eine Folgeerscheinung von Alkoholabhängigkeit, schweren Leberschäden oder Nierenschwäche sein, sowie bei Unterfunktion der Nebennierenrinde und des Hypophysen-(„Hirnanhangsdrüsen-“) Vorderlappens, außerdem bei einer krebsbefallenen Bauchspeicheldrüse auftreten. Typische Symptome reichen von Heißhungerattacken, Unruhe oder Aggressivität und können bis hin zu Krampfanfällen oder einem Schock führen. Je nach ihrem Ausmaß können bei wiederholtem Auftreten von Unterzuckerung Schäden am Gehirn entstehen oder sogar zum Tod führen*.

Es gibt verschiedene Arten der Blutzuckermessung zur Selbstkontrolle:

- Optische Messung

Hier wird das Blut auf ein Testfeld gesogen, welches von außen sichtbar ist. In diesem sind verschiedene Stoffe eingelagert, welche durch eine Reaktion mit Blut eine Farbänderung des Testfeldes hervorrufen. Diese Farbe wird vom Messgerät erfasst und ausgewertet. Die Bestimmung des Blutzuckers erfolgt aus der Dauer und der Stärke der Farbänderung.

- Nicht-invasive Messung

Die nicht-invasive Messmethode zeichnet sich dadurch aus, dass sie verletzungsfrei und ohne Blutentnahme erfolgt. Die Messung erfolgt dabei über eine optische Spektralanalyse des Augenhintergrundes. Dieser ist besonders stark durchblutet und liefert deshalb sehr genaue Werte. Diese Messmethode ist allerdings noch in der Erforschungsphase und wird bisher nicht eingesetzt. Hauptsächlich liegt das daran, dass diese Messgeräte den Gewebezucker (Glukose pro Volumen) und nicht den Blutzucker pro Blutvolumen liefern (vgl. [10]).

- Glukosebestimmung im Urin

Eine Messung des Harnglukosewertes ist ebenfalls möglich, jedoch muss der Blutzuckerwert dafür stark erhöht sein und einen bestimmten Schwellwert überschritten haben. Dieser hängt von der sog. „Nierenschwelle“ ab, das ist die maximale Rückresorptionskapazität (die Wiederaufnahme eines bereits ausgeschiedenen Stoffes durch Zellen oder Gewebe) der Niere von bestimmten Substanzen, in diesem Fall der Glukose. Die Nierenschwelle ist sehr unzuverlässig und störbar. Beispielsweise kann bei Schwangeren die Nierenschwelle auf unter 120mg/dl absinken, während sie bei Gesunden auch über 200mg/dl liegen kann (vgl. [10]), auch durch Krankheiten ist sie veränderbar. Aufgrund der vorhergenannten Gründe gilt diese Messmethode inzwischen als überholt.

- Amperometrische Messung

Da diese Messmethode im vorliegenden Projekt angewendet wird, wird im nachfolgenden Kapitel genauer auf diese eingegangen und auf dieses Kapitel verwiesen.

---

*) Der Tod tritt jedoch meist nicht durch die Unterzuckerung selbst auf, sondern durch die Folgen zunehmender Bewusstseinstrübung, welche zu Verkehrsunfällen oder Stürzen führen kann. Weiterhin ist es möglich, dass der Betroffene aufgrund fehlender Schutzreflexe bei der Bewusstlosigkeit erstickt.

3 Amperometrische Blutglukosemessung

Die Blutzuckermessung dient der Kontrolle des Verlaufs einer Zuckererkrankung (z.B. Diabetes Mellitus). Dazu lernen die Betroffenen, wie sie eine Messung selbst mit den entsprechenden Geräten durchführen können. Man kann den Blutzuckerwert entweder aus dem Blutplasma, das ist der flüssige und zellfreie Teil des Blutes, oder aus dem Vollblut (Gesamtzusammensetzung des Blutes) gewinnen, dabei unterscheiden sich die Messergebnisse etwas, denn der Wert, der aus dem Blutplasma gewonnen wird, ist durchschnittlich elf Prozent höher (vgl. [11]). Ersteres wird jedoch hauptsächlich zur Bestimmung von Diabetes mellitus genutzt und von einem Arzt mit anschließender Laboruntersuchung eingesetzt. Zur Selbstkontrolle wird jedoch das Vollblut genutzt. Hierzu wird ein Blutstropfen aus der Fingerbeere auf einen Teststreifen gegeben und dieser durch das Messgerät ausgewertet. Die Anzeige des Blutglukosewertes erfolgt anschließend auf dem Display.

Abbildung 2: Teststreifen, das Blut wird mit Kapillarwirkung in ein Testfeld gesogen (Pfeil oben)

Bei der amperometrischen Blutzuckermessung wird das aus der Fingerbeere gewonnene Blut durch Kapillarwirkung in ein Testfeld des Messstreifens gesogen, wo es Kontakt zu Glucose-Oxidase (GOD) und den Elektroden des Teststreifens hat. An diesen wird nun eine definierte Spannung gelegt (etwa 300-600mV) wodurch das Blut mit der Glucose-Oxidase reagiert und Gluconsäure gebildet wird. Dabei bildet sich auch Wasserstoff (H_2O_2), der durch die angelegte Spannung in einem elektrochemischen Prozess leicht zersetzt wird und so Elektronen freisetzt. Proportional zur Konzentration entsteht ein messbarer Strom, der, wenn sich die Reaktionen in einem geschlossenen System abspielen, ebenfalls zu der Glukosekonzentration proportional ist.

4 Der Aufbau des Messgeräts

4.1 Messschaltung

Die Schaltung zur Blutglukosemessung, welche im Rahmen dieses Projektes entwickelt wurde, besteht aus mehreren Teilkomponenten. Entsprechend dem in Tabelle 1 dargestellten Farbcode soll im Folgenden auf die genannten Schaltungsteile eingegangen werden. Die dazugehörigen Komponenten sind dementsprechend im Schaltplan auf der folgenden Seite markiert.

Kurzbeschreibung	Farbcode	Thematisierendes Kapitel
IPv6- Modul		4.2
Bildschirm & Touchsensor		4.5
Erzeugung der Messspannung		4.7.1
Messstreifenerkennung		4.7.2
Sensor (Messschaltung)		4.7.3

Tabelle 1: Kapitelübersicht mit Zuordnung zu den einzelnen Schaltungselementen

Weiterhein wurde für den übersichtlicheren Einsatz der Schaltungskomponenten des Blutzuckersensors ein eigenes Bauteil in *Eagle* erstellt. Dazu gehören:

- Das „Symbol“ des Sensors im Schaltplan mit den Bezeichnungen der Pins (Abbildung 3)
- Das „Package“, welches die exakten Abmessungen des Sensors hat und als Bauteil auf dem Layout der Platine eingefügt und genutzt wird (Abbildung 4).

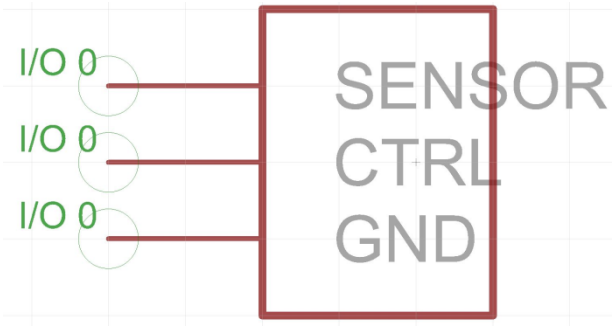


Abbildung 3: Symbol des Sensors

**Abbildung 4: erstelltes Package des BZM;
Kontakte: Oben: Messspannung;
Mitte: Messstreifenkontrolle; Unten: Masse**

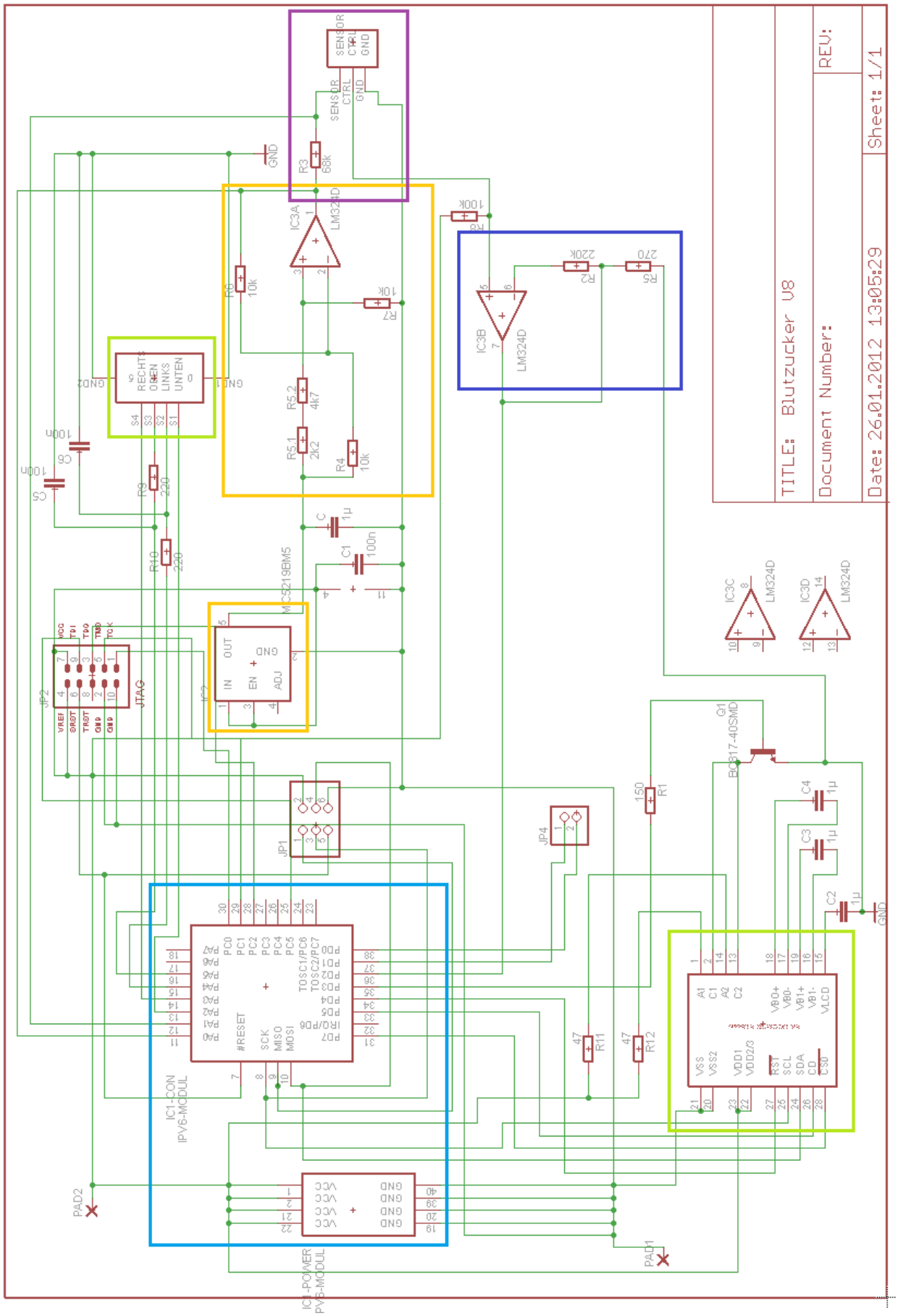


Abbildung 5:Im Rahmen des Projekts entwickelte Blutglukose-Messschaltung

4.2 IPv6 - Funkmodul

Das IP-Protokoll ist auf Schicht 3 des OSI-Referenzmodells, der Vermittlungsschicht (Abbildung 6, gelb hinterlegt), eingegliedert und dient der Übertragung von Daten, welche in Paketen zusammengefasst versendet werden. Das bedeutet, dass es die Grundlage für die Transportprotokolle (TCP, UDP) ist.

OSI-Schicht	TCP/IP-Schicht	Beispiel
Anwendungen (7)	Anwendungen	HTTP, FTP, SMTP, POP, Telnet, OPC UA
Darstellung (6)	(=OSI 5–7)	
Sitzung (5)		SOCKS
Transport (4)	Transport	TCP, UDP, SCTP
Vermittlung (3)	Internet	IP (IPv4, IPv6)
Sicherung (2)	Netzzugang	Ethernet, Token Bus, Token Ring, FDDI
Bitübertragung (1)	(=OSI 1–2)	

Abbildung 6: Das OSI- Referenzmodell. IPv6 befindet sich in Schicht 3 (Vermittlung)

Dabei stellt IPv6 als Protokoll eine Adressierung her, die nicht nur innerhalb eines Subnetzes, sondern über Teilnetze hinweg einmalig und gültig ist. Das macht IPv6 vor allem für die Internettechnologie besonders interessant. Zukünftig wird IPv6 das bisher weiter verbreitete IPv4 ablösen, vor allem, da es etwa 2^{96} mal mehr Adressen bietet als Version 4 des Internetprotokolls, nämlich 2^{128} (das sind etwa $3{,}4 \cdot 10^{38}$). Außerdem wird der Protokollrahmen im Vergleich dazu vereinfacht und verbessert und die Konfiguration von Adressen automatisiert. Des Weiteren wird es aufgrund der Masse an verfügbaren Adressen möglich sein, dass ein Rechner, oder ein beliebiges anderes Gerät mit IPv6-Adresse, diese fest zugewiesen bekommt und so im eigenen Heimnetzwerk wie aber auch im World Wide Web, auf Konferenzschaltungen etc. zu nutzen ist, dies wird *mobile IP* genannt. Eine Implementierung des Sicherheitsprotokolls IPsec ist innerhalb des IPv6-Standards implementiert und macht es möglich, Authentizität und die Verschlüsselungen von IP-Paketen zu überprüfen. Dies wurde bereits bei IPv4 angeboten, die Unterstützung dessen war jedoch lediglich optional. Theoretisch können auch Pakete von bis zu 4GB Größe mit IPv6 versandt werden.

Der IPv6- Header besteht aus mehreren Teilfeldern (siehe Abbildung 7) und dessen Länge ist fest definiert.

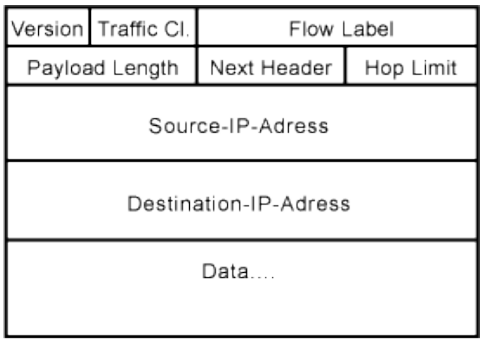


Abbildung 7: Felder des IPv6- Headers

Dies ist eine der Vereinfachungen in IPv6, denn nun befinden sich die jeweiligen Daten immer an derselben Stelle im Datenpaket. Der Router kann so direkt dorthin springen und muss nicht nach den erforderlichen Informationen suchen. Die Quell- und die Zieladressfelder haben jeweils 64 bit („64 bit aligned“), der komplette IPv6-Header ist 40 Byte lang obwohl er weitaus weniger Felder besitzt, als der IPv4 Header (Die Felder IHL, Type of Service, Kennung und Header- Checksumme, die man in diesem findet, wurden aus dem IPv6- Header entfernt).

In der folgenden Tabelle finden Sie eine Übersicht über die einzelnen Felder des IPv6- Headers, deren Länge und eine Beschreibung.

Feldinhalt	Länge/Bit	Beschreibung
Version	4	Hier ist die Version des IP-Protokolls abgelegt, nach der das IP-Paket erstellt wurde.
Traffic Class	8	Der Wert des Feldes definiert die Priorität des Paketes.
Flow Label	20	Das Flow Label kennzeichnet Pakete für ein viel schnelleres Routing. Das MPLS („Multiprotocol Label Switching“) macht dieses Verfahren allerdings überflüssig.
Payload Length	16	Hier stehen die im IP-Paket transportierten Daten in Byte. Bisher musste der Wert aus dem Feld Paketlänge abzüglich dem Feld IHL ermittelt werden.
Next Header	8	Hier ist das übergeordnete Transportprotokoll angegeben. Bei IPv4 hieß das Feld einfach Protokoll.
Hop Limit / TTL	8	Dieses Feld enthält die Anzahl der verbleibenden weiterleitenden Stationen, bevor das IP-Paket verfällt. Es entspricht dem TTL-Feld von IPv4. Jede Station, die ein IP-Paket weiterleitet, muss von diesem Wert 1 abziehen.
Source-Adress	128	In der Source-Adress steht die IP-Adresse der Station, die das Paket abgeschickt hat (Quell-IP-Adresse).
Destination-Adress	128	An dieser Stelle steht die IP-Adresse der Station, für die das Paket bestimmt ist (Ziel-IP-Adresse).
IPv6-Header-Erweiterungen	jeweils 64 Bit (8 Byte)	Im IPv6-Header können optional Informationen im separaten Header dem IP-Kopf angehängt werden. Bis auf wenige Ausnahmen werden diese Header-Erweiterungen von IP-Routern nicht beachtet.

Tabelle 2: Beschreibung der einzelnen Felder im IPv6- Header (vgl. [6])

Integriert in das verwendete Funkmodul ist ein Mikrocontroller der Firma *Atmel*, der *ATmega1284P*, von dem eine Schemazeichnung mit den Ein- und Ausgängen in Abbildung 8 auf der nachfolgenden Seite zu sehen ist. Für die Ansteuerung über SPI („Serial Peripheral Interface“) sind die Ports PB5 (MOSI: „Master Out Slave In“), PB6 (MISO: „Master In Slave Out“), PB4 (SS: „Slave Select“) und PB7 (SCK: „Serial Clock“) (siehe Kapitel 4.3) vordefiniert. Des Weiteren verfügt dieser Mikrocontroller über einen 16kByte Speicher, sowie über einen 10 Bit Analog- Digital- Converter an Port A (mit PA0 … PA7: ADC0 … ADC7) (für Weiteres siehe Abschnitt 4.4).

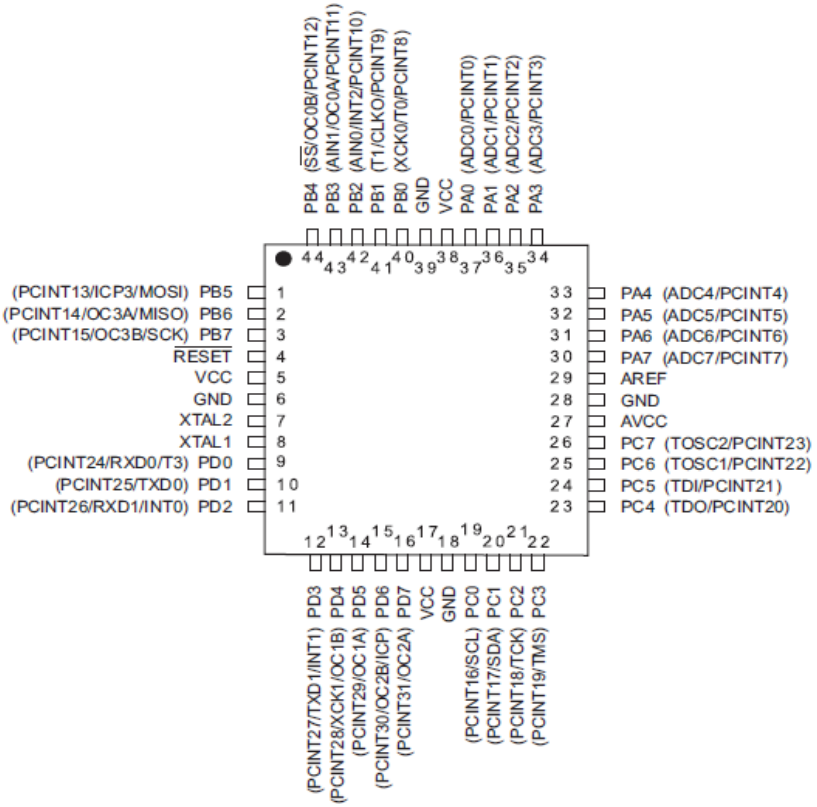


Abbildung 8: Schemazeichnung ATmega 1284P mit Bezeichnung der Ein- und Ausgänge

4.3 Serial-Peripheral-Interface (SPI)

4.3.1 Grundlagen

Das Serial- Peripheral- Interface, kurz SPI, ist ein von Motorola entwickelter, sehr offener, lizenzfreier Standard für einen synchronen seriellen Datenbus, welcher nach dem Master- Slave- Prinzip arbeitet und keinen verpflichtenden Normen unterliegt. SPI ist vollduplexfähig, das bedeutet, dass die Kommunikation gleichzeitig in beide Richtungen, also vom Master zum Slave und umgekehrt, erfolgen kann. Dies ist auch zwingend notwendig, da bei SPI immer genau ein Byte übertragen wird (auch bei 16- oder 32-Bitsystemen) und nur ein Register für Sende- und Empfangsdaten vorgesehen ist. Wird also vom Master ein Byte gesendet, wird zum gleichen Zeitpunkt ein Byte vom Slave an den Master gesandt. Für die Übertragung eines Bytes werden acht Taktzyklen gebraucht, da pro Periode nur ein Bit übertragen werden kann. Allerdings ist es möglich, mehrere Bytes direkt hintereinander zu übertragen. Die Signalisierung, dass eine Übertragung beendet ist, erfolgt durch das Setzen des „Slave Select Signals“ (dazu später mehr).

Bei SPI kann man viele Einstellungen vornehmen, wie zum Beispiel die Wortlänge, ob zuerst das MSB („Most Significant Bit“, das höchstwertigste Bit entspricht bei einem Byte dem Wert Low oder High für die Stelle 2^7) oder das LSB („Least Significant Bit“, entspricht im Byte der Stelle 2^0) gesendet werden soll, aber auch die Taktflanke, mit der ausgegeben werden soll (steigende oder fallende Flanke). Weitere Vorteile von SPI sind, dass man beliebig viele Slaves anschließen kann (es darf allerdings nur genau einen Master geben!). Des Weiteren ist es möglich, dass Mikrocontroller, die einen integrierten Flashspeicher haben, wie auch der in diesem Projekt eingesetzte ATmega1284P, programmiert werden können, ohne dass sie aus der Schaltung entfernt werden müssen („In-System-Programming“, ISP). Auf die gleiche Weise lässt sich der Speicher auch auslesen, hierzu sind lediglich die entsprechenden Steckplätze auf der Platine vorzusehen und zu verdrahten.

Im vorliegenden Schaltplan sind drei Slaves an den Master (den Mikrocontroller) angeschlossen, nämlich zum einen der Touchscreen und das IPv6- Modul und zum anderen die Programmierschnittstelle. Dies ist möglich, da der Master mit Hilfe des Slave Select Signals in der Lage ist, einzelne Slaves gezielt anzusprechen. Pro Slave ist eine Leitung vorgesehen, welche im Normalfall „Low-Active“ ist, das bedeutet, dass sie im Ruhezustand auf logisch 1 gesetzt ist (alternativ finden sich je nach Anwendung die Bezeichnungen „Slave Select“ (SS), „Chip Select“ (CS) oder zum Beispiel „Slave Transmit Enable“ (STE)). In Abbildung 9 sehen Sie ein schematisches Beispiel für einen Fall, in dem mehrere Slaves an den Master angeschlossen sind. Wird der Slave über dieses Signal selektiert, „lauscht“ er an der MOSI- Leitung („Master out, Slave in“, die

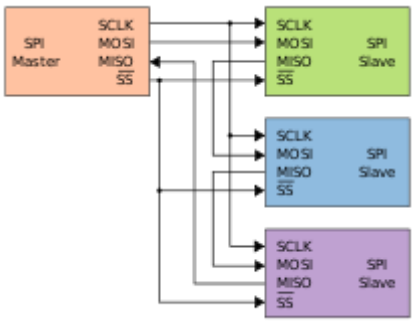


Abbildung 9: Mehrere Slaves können über SPI mit dem Master verbunden werden

Datenbits, welche an den Slave übergeben werden sollen) und legt die Daten in seinem Senderegister im Takt des vom Master bereitgestellten und generierten Clocksignals (SCK, Transporttakt für die Datenbits) an die MISO-Leitung („Master in, Slave out“, die ausgelesenen Daten) an.

Zur Programmierung über die SPI-Schnittstelle ist es nötig, einige Einstellungen bzw. Definitionen („Flags“) des „SPI-Control-Registers“, kurz SPCR im Programm des Mikrocontrollers festzulegen, diese sind in den folgenden Tabellen dargestellt:

Bit	7	6	5	4	3	2	1	0
SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Initialwert	0	0	0	0	0	0	0	0

Tabelle 3: SPCR - SPI- Control- Register

Name des Flags	Kurzbezeichnung	Beschreibung
SPI-Interrupt Enable	SPIE	Logisch 1, wenn der SPI- Interrupt aktiviert ist, logisch 0, wenn er deaktiviert ist
SPI Enable	SPE	Logisch 1, wenn SPI angeschaltet ist, logisch 0, wenn es abgeschaltet ist
Data Order	DORD	Beschreibt, welches Bit im Datenregister zuerst gesendet werden soll. Bei logisch 0: MSB, bei logisch 1: LSB
Master/Slave Select	MSTR	Beschreibt, ob der Mikrocontroller als Master oder Slave fungiert, logisch 1: Controller entspricht dem Master
Idle Polarity	CPOL	Polarität der Taktleitung im Ruhezustand
Clock Phase Bit	CPHA	Beschreibung, bei welcher Taktflanke die Daten übernommen werden sollen
SPI Clock Rate Select	SPR1/SPR0	Legen die Taktrate des SCK-Signals in Abhängigkeit vom CPU-Takt fest (nur, wenn der Mikrocontroller als Master fungiert)

Tabelle 4: Übersicht und Beschreibung der Flags bei SPI

Das Serial Peripheral Interface bietet vielfältige Einsatzmöglichkeiten, beispielsweise in der Datenübertragung zwischen Mikrocontrollern, bei Messanwendungen oder sogar im Audiobereich, allerdings ist es ungeeignet für die Anbindung von externen Komponenten, da es keinerlei Störschutz gegen externe Signale bietet.

4.3.2 Anwendung des SPI im Programmcode

Die Pin-/ bzw. Portbelegung für den SPI in der entwickelten Schaltung ist folgende:

Bezeichnung	DDR/Port	Anschluss	Wert (High = 1/Low = 0)
MISO	B	6	0
MOSI	B	5	1
SCK	B	7	1
SS	B	4	1

Tabelle 5: Pin-/Portbelegung für den SPI in der Schaltung

An das SPI sind zum einen der Touchscreen und zum anderen das Funkmodul und die Programmierschnittstellte angeschlossen. Der Mikrocontroller fungiert als Master, woraus sich auch die folgenden Bitbelegungen für die Initialisierung des Serial Peripheral Interface im SPCR-Register ergeben (die Bedeutung der einzelnen Bits und der zugehörigen logischen Zustände befindet sich in Kapitel 4.3.1: „Grundlagen“):

Bit	7	6	5	4	3	2	1	0
SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Zuweisungswert	0	1	0	1	0	0	0	0

Tabelle 6: Zuweisung der Werte des SPCR- Registers im Projekt

CPOL und CPHA sind beide auf logisch 0 gesetzt, dies entspricht beim hier eingesetzten Mikrocontroller dem SPI-Mode 0. Das bedeutet, dass die Daten bei einer steigenden Taktflanke übernommen werden (siehe Abbildung 10 mit CPOL = 0).

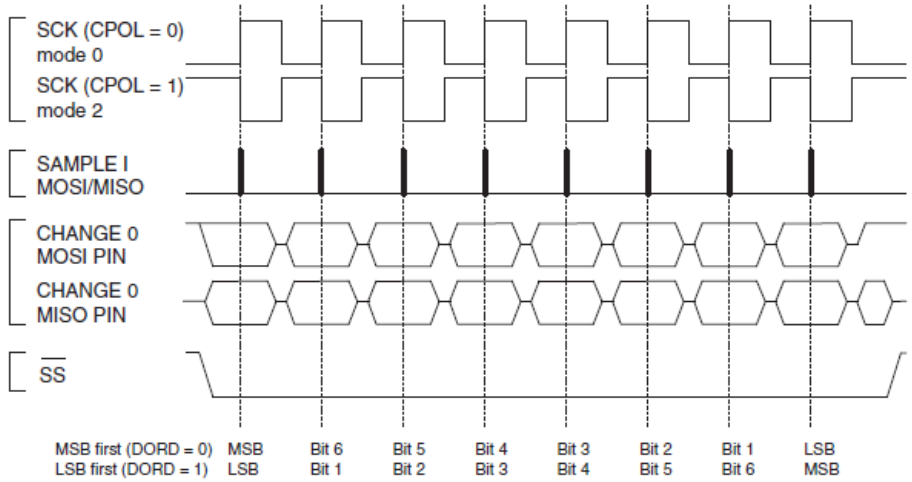


Abbildung 10: SPI- Übertragung mit CPHA=0

Mit der Einstellung SPR1 und SPR0 auf logisch 0, wird das Verhältnis zwischen SCK und Oszillatortakt gesetzt auf:

$$f_{sck} = \frac{f_{osc}}{4}$$

Diese Zuordnungen werden in der Funktion init_SPI() definiert, welche am Anfang des Hauptprogrammes über Init_all aufgerufen wird und maßgeblich für die Kommunikation zwischen Master und Slaves ist.

Soll nun eine Anweisung bzw. ein Datenstrom über den SPI gesendet werden, geschieht dies mit der Funktion spi_sende(unsigned char spi_sendebyte). Man übergibt einen 8-Bit Wert, welcher dann über das Shift-Register (SPDR) übertragen wird.

Mit der Zeile

```
while (!(SPSR & (1<<SPIF)));
```

wird das Ende der Übertragung abgewartet, dann wird SPIF, das „SPI Interrupt Flag“ aus dem „SPI-Status-Register“ (SPSR) auf logisch 1 gesetzt und die Bedingung für die while-Schleife ist nicht mehr erfüllt. Folglich könnte nun eine weitere Übertragung gestartet werden. Alle Funktionen, die die Nutzung des SPI betreffen, waren bereits als Programmcode vorhanden und mussten nur noch an das Projekt angepasst werden.

4.4 Analog- Digital – Wandler

4.4.1 Allgemeine Funktionalität

Grundsätzlich besteht ein A/D-Wandler aus analogen Eingängen, einer Referenzspannung und digitalen Ausgängen. Seine Aufgabe ist die Umsetzung von analogen Eingangswerten in digitale Ausgangswerte, die relativ zur Referenzspannung berechnet werden. Letztere repräsentiert gleichfalls den oberen Grenzwert (und den unteren bei vorzeichenbehafteter Umwandlung). Dabei gibt es die Möglichkeit, entweder nur einen Pin (Single- Ended- Input) oder zwei Pins einzulesen und die Differenz derer als Digitalcode auszugeben (Differential-Conversion- Mode).

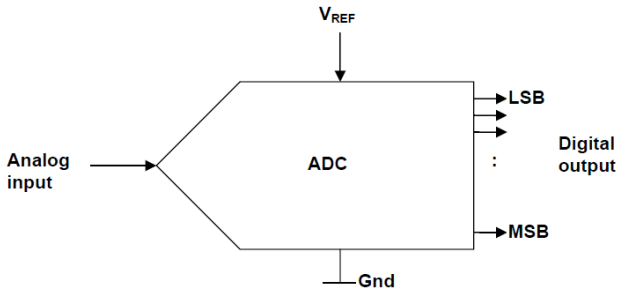


Abbildung 11: Vereinfachte Darstellung des A/D-Wandlers

Die wichtigsten Grundbegriffe sind:

- Eingangsspannungsbereich (Input-voltage range)

Der mögliche Eingangsspannungsbereich wird bestimmt durch die anliegende Referenzspannung VREF, die entweder intern oder extern an einen Pin des Mikrocontrollers angeschlossen vorhanden sein muss. Ob VREF intern oder extern vorliegt, kann mit Setzen und Löschen der entsprechenden Bits eingestellt werden (dazu später mehr). Die Referenzspannung entspricht der höchsten umwandelbaren Spannung und somit den höchsten digitalen Code.

Werden Spannungen angelegt, deren Betrag größer ist als VREF wird der ADC übersättigt und gibt den größten darstellbaren Code aus.

- Auflösung (resolution)

Die kleinste Änderung des Ausgangs am A/D- Wandlers (genannt „LSB“) bezeichnet man als Auflösung. Damit eng verknüpft ist die maximale Codewortlänge. Hat ein ADC beispielsweise die Auflösung 8 entspricht das einer Wortlänge von drei Bit ($8=2^3$).

- Quantisierung

Die Umwandlung von analogen zu digitalen Werten erfolgt im Idealfall stufenlos (siehe Abbildung 12 links). Man bräuchte allerdings eine unendlich hohe Auflösung, um eine derartige Wandlung vorzunehmen, deshalb erfolgt im Realfall eine Quantisierung des Messwertes (siehe Abbildung 12 rechts). Dabei werden Spannungsstufen mit Werten codiert, so dass ein Fehler entsteht, der proportional zur Breite der Quantisierungsstufen ist.

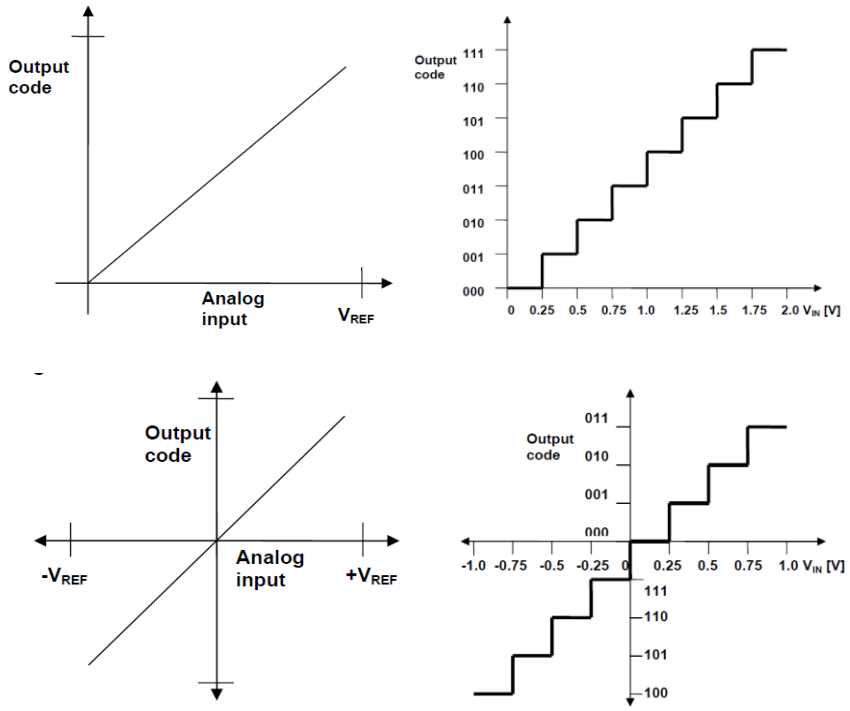


Abbildung 12: links: idealer Ausgang des ADC, rechts: Quantisierung der analogen Werte, oben: Single- Ended- Input, unten: Differential- Conversion- Mode

- Umwandlungsmodus („Conversion Mode“)

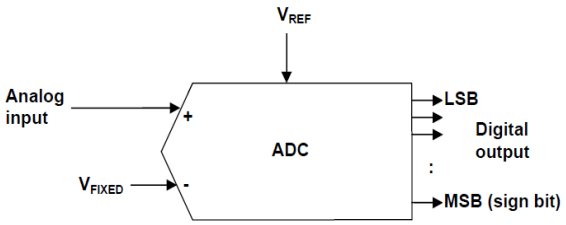


Abbildung 13: A/D- Wandler im "Unsigned Mode"

Im *Single- Ended- Mode* wird nur ein einzelner Analogeingang in einen digitalen Code umgesetzt. Dazu wird dieser auf den positiven Eingang des Wandlers gelegt, während der negative Eingang entweder auf Massepotenzial (wenn das Vorzeichen des Analogwertes für die Weiterverarbeitung benötigt wird; „signed mode“) oder auf ein festes Potenzial V_{FIXED} (wenn das Vorzeichen des Messwertes nicht benötigt wird; „unsigned mode“). Das Vorzeichen wird über das höchstwertigste Bit ausgegeben.

Der *Differential- Conversion- Mode* ist für die Berechnung der Differenz zweier Eingangswerte und der nachfolgenden Umwandlung derer in einen Digitalcode vonnöten. Diese werden auf den positiven und den negativen Eingang des ADC gelegt, je nach Anwendungsfall kann man hier einen programmierbaren Verstärker zwischenschalten (Abbildung 14). Dieser Modus wird meist

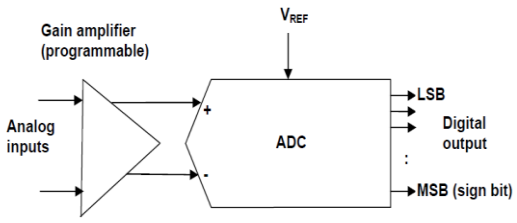


Abbildung 14: ADC im Diffential- Conversion- Mode, vorzeichenbehaftet

vorzeichenbehaftet eingesetzt, wobei auch hier jenes durch das MSB ausgedrückt wird. Der Umwandlungsbereich liegt hier durch den „signed- Mode“ bei $\pm VREF$.

- Parameter

Ein Analog- Digital- Wandler hat verschiedene Parameter, die sich in statische und dynamische aufteilen. Die ersteren haben keinen Bezug zum Eingangssignal und sind somit davon unabhängig. Dazu gehören Offset-, Verstärkungs-, Linearitäts und Maßstabsfehler. Dynamische Parameter sind hingegen vom Eingangssignal abhängig und deren Einfluss steigt mit der Frequenz. Zu diesen gehört beispielweise das SNR(„signal- to- noise- ratio“) des A/D-Wandlers.

- Timings

Im ADC treten einige Verzögerungs- und Wandlungszeiten auf, die für gewöhnlich in Clock-Raten gemessen werden. Anlauf-, Abtast-, Halte- und Einschwingzeit sind in modernen Mikrocontrollern per Software durch Setzen oder Löschen bestimmter Bits konfigurierbar.

Bezeichnung	Beschreibung
Anlaufzeit (Startup- Time)	Minimale Zeit, die benötigt wird, damit die maximale Auflösung des ADC genutzt werden kann (vor der ersten Wandlung oder nach dem Aufwachen aus dem Sleep-Modus).
Abtast- & Haltezeit (Sample & Hold- Time)	Sample: Nach Starten der Wandlung muss zuerst der interne Kondensator auf einen stabilen Wert gebracht werden, bevor ein präziser Wert verfügbar wird Hold: Zeit, die für die interne Konvertierung der Spannung in einen entsprechenden Digitalwert benötigt wird.
Einschwingzeit (Settling- Time)	Werden verschiedene Kanäle mit unterschiedlichen Verstärkungs- und Offsetkonfigurationen genutzt, kann das Wechseln dieser einige Zeit in Anspruch nehmen, bevor die Sample & Hold- Phase begonnen werden kann. Eine A/D Umwandlung sollte niemals vor Ablauf der Einschwingzeit erfolgen, da dies die Genauigkeit erheblich verringert.
Umwandlungszeit (Conversion- Time)	Kombination aus Abtast- und Haltezeit. Wesentlicher Faktor, der die Geschwindigkeit des Analog- Digital-Wandlers bestimmt.

Tabelle 7: Beschreibung der verschiedenen Timings im ADC

- Sampling- Rate und Bandbreite

Als Sampling- Rate bezeichnet man die Anzahl der möglichen Abtastungen pro Sekunde, während die Bandbreite die maximale Frequenz darstellt, die dem A/D- Wandler zugeführt werden kann. In Anlehnung an das Nyquist- Kriterium sollte die Sampling- Rate mindestens doppelt so hoch sein, wie die Bandbreite des Eingangssignals.

Um den ADC zu konfigurieren, müssen Bits in den entsprechenden *Registern* gesetzt werden. Im Folgenden wird auf die im Projekt benötigten Registern des A/D-Wandlers näher eingegangen:

- ADCSRA – Register (ADC Control- and Status- Register A)

Bit	7	6	5	4	3	2	1	0
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Initialwert	0	0	0	0	0	0	0	0

Tabelle 8: Initialwerte des ADCSRA- Registers

Name des Flags	Kurzbezeichnung	Beschreibung
ADC Enable	ADEN	Logisch1: ADC einschalten; Logisch 0: ADC ausschalten
ADC Start Conversion	ADSC	Logisch 1: Beginn der Wandlung; Bit bleibt so lange auf 1, bis die Wandlung abgeschlossen ist, danach wird das Bit von der Hardware auf logisch 0 gesetzt
ADC Auto Trigger Enable	ADATE	Logisch 1: Bei einer positiven Flanke des ausgewählten Triggersignals wird automatisch eine Wandlung gestartet
ADC Interrupt Flag	ADIF	Wird nach Abschluss einer Messung auf logisch 1 gesetzt. Zusätzlich dazu wird das ADIE- Bit auf logisch 1 gesetzt
ADC Interrupt Enable	ADIE	Interrupt- Auslösung nach Setzen dieses Bits
ADC Prescaler	ADPS2/ADPS1/ ADPS0	Wählen der ADC- Frequenz (siehe Datenblatt des Herstellers)

Tabelle 9: Bedeutung der Bits im ADCSRA- Register

- ADMUX – Register (ADC Multiplexer Selection Register)

Bit	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Initialwert	0	0	0	0	0	0	0	0

Tabelle 10: Initialwerte des ADMUX- Registers

Name des Flags	Kurzbezeichnung	Beschreibung
Reference Selection Bits	REFS1/REFS0	Auswahl der Referenzspannung, (extern oder intern)
ADC Left Adjust Result	ADLAR	Bestimmt die Ausrichtung der Ergebnisbits im ADCH/ADCL- Register
Analog Channel and Gain Selection Bits	MUX4…MUX0	Wahl der Verstärkung und der Kanäle für die Wandlung (siehe Datenblatt des Herstellers)

Tabelle 11: Bedeutung der Bits im ADMUX- Register

- ADCL- und ADCH – Register (ADC Data Register)

Je nach Einstellung des ADLAR- Bits wird das Ergebnis entweder links- oder rechtsbündig ausgegeben. Daraus ergeben sich folgende Registerbelegungen (ADC0…ADC9 entsprechen den Bits des umgewandelten Digitalwerts):

Bit	7	6	5	4	3	2	1	0
ADCH	-	-	-	-	-	-	ADC9	ADC8
Initialwert	0	0	0	0	0	0	0	0
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Initialwert	0	0	0	0	0	0	0	0

Tabelle 12: Initialwerte des ADCH- und ADCL – Registers mit ADLAR = 0, Ausrichtung rechtsbündig

Bit	7	6	5	4	3	2	1	0
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
Initialwert	0	0	0	0	0	0	0	0
ADCL	ADC1	ADC0	-	-	-	-	-	-
Initialwert	0	0	0	0	0	0	0	0

Tabelle 13: Initialwerte des ADCH- und ADCL – Registers mit ADLAR = 1, Ausrichtung linksbündig

4.4.2 Nutzung des A/D- Wandlers im Projekt

Der A/D- Wandler wird im vorliegenden Projekt an zwei Stellen verwendet:

- Zur Bestimmung der Koordinaten des Berührpunktes auf dem Touchscreen und
- Zur Messung des Blutzuckerwertes

Die Koordinaten des Berührpunktes auf dem Touchsensor werden jeweils für die Abszisse und die Ordinate getrennt aufgenommen. Dies erfordert zwei Messungen im Single- ended- Mode. Entsprechend der Hardware wird der X-Wert über Pin A3 und der Y- Wert über Pin A2 ausgelesen und in einen entsprechenden Digitalwert umgewandelt.

Aus der Schaltung und der innerhalb des Programmcodes gewünschten Konfiguration können folgende Einstellungen über die entsprechenden Bits gemacht werden:

- Die Versorgungsspannung AVCC wird extern über einen Kondensator zugeführt
    - REFS1 = 0; REFS0 = 1
- ADC-Kanal für die Bestimmung der X- Koordinate: PA3 (entspricht „Oben“)
    - MUX0…MUX1 = 1; MUX2…MUX4 = 0
- ADC-Kanal für die Bestimmung der Y- Koordinate: PA2 (entspricht „Links“)
    - MUX1 = 1, MUX0 = MUX2…MUX4 = 0
- Teilungsfaktor zwischen XTAL und ADC- Clock: 64
    - ADPS2 = ADPS1 = 1
- Verstärkung: keine (Single- ended- Mode)

Somit wurden folgende Bitzuweisungen vorgenommen:

Bestimmung der X- Koordinate:

Bit	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Zuweisungswert	0	1	0	0	0	0	1	1

Tabelle 14: Zuweisung der Werte des ADMUX- Registers zur Bestimmung der X-Koordinate

Bestimmung der Y- Koordinate:

Bit	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Zuweisungswert	0	1	0	0	0	0	1	0

Tabelle 15: Zuweisung der Werte des ADMUX- Registers zur Bestimmung der Y-Koordinate

Für die Ermittlung beider Werte sind die Bitpositionen im ADCSRA- Register gesetzt zu:

Bit	7	6	5	4	3	2	1	0
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Zuweisungswert	1	0/1	0	0	0	1	1	0

Tabelle 16: Zuweisung der Werte des ADCSRA- Registers

Für die Bestimmung des Blutzuckermesswertes ist eine Differenzen-Auswertung mit Hilfe des A/D-Wandlers vonnöten, da ein Spannungsabfall über einem Widerstand gemessen wird. Zu beachten ist, dass der ADC hierzu zwar an verschiedenen Stellen im Programmcode genutzt wird (zur Bestimmung des Messwertes und zur Erkennung des Blutes auf dem Messstreifen), doch bleiben für beide Fälle die Einstellungen gleich, da die Messung jeweils über Pin 1 und Pin 0 erfolgen.

Hard- und Softwaretechnisch sind folgende Einstellungen umzusetzen:

- Die Versorgungsspannung AVCC wird extern über einen Kondensator zugeführt (s.o.)
    - REFS1 = 0; REFS0 = 1
- Differenzenmessung über Pin 1 und Pin 0 mit Verstärkungsfaktor 10
    - MUX3 = MUX0 = 1
- Teilungsfaktor zwischen XTAL und ADC- Clock: 64
    - ADPS2 = ADPS1 = 1

Somit müssen folgende Bitzuweisungen vorgenommen werden:

Bit	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Zuweisungswert	0	1	0	0	1	0	0	1

Tabelle 17: Zuweisung der Werte des ADMUX- Registers

Bit	7	6	5	4	3	2	1	0
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Zuweisungswert	1	0/1	0	0	0	1	1	0

Tabelle 18: Zuweisung der Werte des ADCSRA- Registers

Jede Messung wird durch Setzen des ADSC- Bits im ADCSRA- Register gestartet, welches nach einer beendeten Messung durch die Hardware zurück auf logisch 0 gesetzt wird.

Die Zeile

```
while (ADCSRA & (1<<ADSC));
```

wartet dieses Ende der Wandlung ab, um fehlerhafte Digitalcodes zu vermeiden. Der ausgelesene und umgewandelte Analogwert wird daraufhin in einer Variable gespeichert und weiterverarbeitet.

Näheres zur Bestimmung der Koordinaten des Berührungspunktes auf dem Touchfeld befindet sich in Kapitel 4.5.2 „Auswertung des Touchsensors im Projekt“.

Für weitere Informationen zum Thema Messwerterfassung und –auswertung wird auf die Kapitel 4.7.2 „Messstreifenerkennung“ und 4.7.3 „Auswertung des Messwertes“ verwiesen.

4.5 Bildschirm und Touchsensor

4.5.1 Allgemeine Funktionalität eines Touchsensors

Die sogenannten Touchscreens (auch Tastschirm, Sensorbildschirm oder Berührungsbildschirm genannt) erfreuen sich immer größerer Beliebtheit, sei es bei Automaten, Smartphones oder Tablets, um nur ein paar Anwendungsgebiete zu nennen. Dies liegt vor allem an der stark vereinfachten Bedienbarkeit. Es ist nicht mehr nötig, umständlich über Eingabegeräte, wie Maus oder Tastatur auf einem Bildschirm „herumzuklicken“. Da die Benutzereingaben bei einem Touchscreen über den Bildschirm selbst erfolgen, ist dieser zugleich Ein- und Ausgabemedium.

Abbildung 15: Dr. G. S. Hurst (links), Gründer von "Elographics" präsentiert den ersten Touchscreen ("Elograph")

Der erste Sensorbildschirm wurde bereits 1971 von Dr. Sam Hurst, dem Gründer von „Elographics“ (heute „Elo Touchsystems“) bekannt. Der „Elograph“ war nicht transparent, aber diente trotzdem als Grundlage für die heutigen Touchscreens. Der erste Berührungsbildschirm mit einer solchen, transparenten Oberfläche wurde ebenfalls von Hurst im Jahre 1974 entwickelt und vorgestellt.

Im folgenden Abschnitt soll etwas genauer auf die Bestandteile eines solchen Touchscreens eingegangen werden. Den Bildschirm an sich einmal außen vor gelassen, besteht ein solcher aus drei Teilen:

- Dem *Touchsensor*, der in der Regel eine berührungsempfindliche Oberfläche aus Glas oder flexiblem Polyester ist. Die am weitesten verbreitete Technik zur Ermittlung der Position auf einem Touchscreen ist die, dass ein Strom über die Sensoroberfläche fließt, was bei einer Berührung einen Spannungswechsel hervorruft (dazu später mehr).
- Dem *Controller*, welcher im Allgemeinen mit dem Monitor selbst verbaut ist und der die Aufgabe hat, die Benutzereingaben auf dem Sensor zu erfassen, damit diese ausgewertet werden können.
- Und zuletzt dem *Softwaretreiber*. Diese Software dient der Interpretation der vom Kontroller gelieferten Signale und muss auf dem System installiert sein, welches die Daten verarbeiten soll. Meist werden durch Mausemulatoren die Berührungen auf dem Touchscreens als Mausklick an eben jene Stelle simuliert.

Es gibt eine große Anzahl verschiedener Sensorbildschirme, die auf unterschiedlichen Techniken beruhen: kapazitive, induktive, resistive Touchscreens und noch viele mehr. Im Aufbau des in diesem Projekt entwickelten Blutzuckermessgerätes findet sich ein resistiver Touchscreen, weshalb im Folgenden diese näher erläutert werden sollen.

Bei resistiven Touchscreens wird eine geringe Prüfspannung auf die zwei gegenüberliegenden, leitfähigen Schichten aus Indiumzinnoxid (ITO), einem lichtdurchlässigen Halbleiter, gegeben. Die hintere Schicht liegt auf einer stabilen Oberfläche, die obere Schicht wird mit sogenannten „Spacer-Dots“ von der unteren im berührungslosen Zustand gehalten, solange keine Bedienereingabe am Touchfeld erfolgt und ist flexibel und druckempfindlich. Mit Hilfe der Spacer-Dots fließt also kein Ruhestrom. Wird die Oberfläche der oberen Ebene jedoch berührt, werden die elektrisch leitfähigen Schichten durch den Druck stellenweise verbunden (siehe Abbildung 16), ein Widerstand entsteht, durch den es möglich wird, die Position der Eingabe zu berechnen. Dies geschieht dadurch, dass durch den elektrischen Kontakt der Schichten ein Spannungsteiler entsteht.

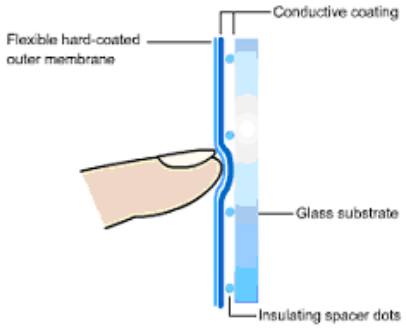


Abbildung 16: Bei Druck werden die elektrisch leitfähigen Schichten miteinander verbunden

Am häufigsten anzutreffen sind die four- bzw. die five-wire- Touchscreens. Diese bedienen sich ähnlicher Auswertetechniken, die im Folgenden näher erläutert werden sollen.

Der *four- wire- Touchscreen* ist die wohl simpelste Art der Berührungsmessung. Es gibt vier Kanäle als Verbindungen zum Kontroller und beide Oberflächen des Sensors werden für die Bestimmung des Berührungspunktes herangezogen. Auf jeder dieser Schichten gibt es nun zwei Anschlüsse, einen für die Spannung und einen zum Potentialausgleich (Erdung), dabei unterscheiden sich nicht die Drähte von Schicht zu Schicht, sondern deren Orientierung auf dem Sensor (siehe Abbildung 17). Der Messablauf folgt nun in zwei Schritten. Zuerst wird auf hintere Schicht eine definierte Gleichspannung gegeben. Diese Spannung nimmt ab, je weiter sich der Berührungspunkt von der Ursprungskante entfernt. Durch Zusammendrücken der Schichten entsteht ein Spannungsteiler, dessen Verhältnis von der (eindimensionalen) Position des Berührungspunktes abhängt. Der Wert, der nun an den Kontroller geliefert wird, entspricht der x-Koordinate. Daraufhin wird der Prozess umgekehrt: Nun wird die obere Schicht mit einer Gleichspannung gespeist und die Messung erfolgt über die hintere Schicht, dieser Wert entspricht der y-Koordinate. Anzumerken ist hierbei, dass bei diesem System zu jedem Zeitpunkt lediglich drei Leitungen aktiv sind: Erdung, Messung und Speisespannung. Nachteilig wirkt sich allerdings die mechanische Belastung der äußeren Polyesterschicht des Touchscreens aus, denn dadurch verliert die leitfähige Beschichtung an der Innenseite ihre Gleichmäßigkeit und die Präzision bei der Messung des Druckpunktes nimmt ab.

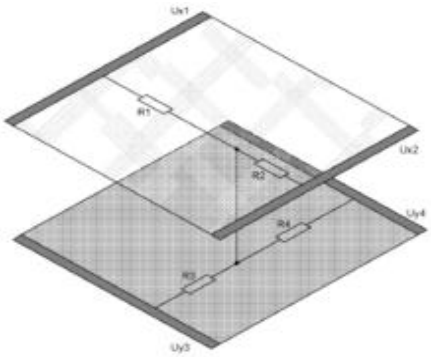


Abbildung 17: Schematischer Aufbau eines 4-wire-Touchscreens

Dieser Nachteil wird beim *five- wire- Touchscreen* dadurch vermieden, dass die äußere Schicht nicht als Sensor genutzt wird. Diese leitet nur die Spannung von der unteren Schicht weiter und ist mit einem

zusätzlichen Draht angeschlossen. Die anderen vier Drähte liegen an den Eckpunkten der unteren Schicht und werden für die Messung herangezogen. Die jeweils benachbarten Ecken werden vor jeder der beiden erforderlichen Messungen direkt miteinander verbunden und dann an diese Eckenpaare die Spannung angelegt. Da im Rahmen dieses Projektes allerdings ein four-wire-Berührungsbildschirm eingesetzt wurde, dessen Berührungsempfindlichkeit für diese Aufgabe völlig ausreichend ist, wird an dieser Stelle auf die entsprechende Fachliteratur verwiesen.

4.5.2 Auswertung des Touchsensors im Projekt

Der Touchsensor ist über den Analog- Digital -Wandler an Port A angeschlossen.

So ist es möglich, die Positionswerte, an denen eine Eingabe per Druck auf den Sensor erfolgte, zu erfassen, welche dann vom Mikrocontroller ausgewertet werden. Die Anschlüsse des Touchsensors wurden zur besseren Veranschaulichung „Oben“, „Unten“, „Rechts“ und „Links“ genannt (siehe Schaltplan, Kapitel 4.1). Für die Bestimmung der X- und Y- Koordinate sind zwei Messungen nötig, jeweils ändert sich auch der Zuweisungswert der jeweiligen Anschlüsse (logisch 0 oder logisch 1).

Bezeichnung	DDR/Port	Anschluss	Wert (High = 1/Low = 0)	
Oben	A	3	X: 0	Y:0
Unten	A	0	X: 0	Y:1
Rechts	A	1	X: 1	Y:0
Links	A	2	X: 0	Y:0

Tabelle 19: Zuweisung der Bitwerte (X: zur Bestimmung der X-Koordinate, Y: zur Bestimmung der Y-Koordinate)

Die Auswertung der Sensorübergabewerte muss an den jeweiligen Bildschirm angepasst werden. Dies geschieht durch Definition der maximalen und minimalen Werte, die über den Sensor ausgegeben werden, wenn man diesen jeweils am untersten und obersten Punkt des Bildschirms in vertikaler und horizontaler Richtung berührt. Diese Grenzen sind nur durch Messen ermittelbar und unterscheiden sich von Bildschirmmodul zu Bildschirmmodul mitunter erheblich. Weiterhin fließen sie in die Auswertung als Erfassungsgrenzwerte mit ein, da die Berechnung der Koordinaten in Prozent erfolgt.

Achse	Minimalwert	Maximalwert
X (Abszisse)	32	218
Y (Ordinate)	56	212

Tabelle 20: Oberer und unterer Grenzwert der auswertbaren Achsen

4.5.3 Funktion und Ansteuerung des Bildschirms im Projekt

Bei dem im Projekt eingesetzten Display handelt es sich um einen *EA DOGS102-6* mit zusätzlicher weißer Hintergrundbeleuchtung (Artikelbezeichnung: EA LED39X41-W) von *Electronic Assembly*. Dieses Bildschirmmodul zeichnet sich vor allem dadurch aus, dass es für den Betrieb an 3,3V (wie im vorliegenden Projekt) keine weitere Spannungsversorgung benötigt und somit sofort in diesem Bereich einsetzbar ist. Der Bildschirm misst 102 x 64 Bildpunkte und wird unterteilt in 8 sog. „Pages“ (Page 0 bis Page 7, siehe Abbildung 18). Im Projekt wurde das Display zur besseren Lesbarkeit mit der Einbaulage 12:00 („Top View“) vorgesehen. Es ist ebenso möglich, die Darstellungen auf diesem umgedreht vorzusehen. Dazu müsste die Einbaulage 6:00 („Bottom View“) gewählt und entsprechend programmiert werden. Je nachdem, wie der Bildschirm montiert wurde und welche Einbaulage gewünscht ist, verändert sich die Unterteilung innerhalb der Pages dahingehend, dass die Zeilenadresse des ersten Bits pro Page verschoben ist. Wie in Abbildung 18 dargestellt, beginnt die Zeilenadressierung im Projekt bei Zeile 30 und endet bei Zeile 131 (Bei Top View: Zeile 0 bis 101), dies entspricht einer Darstellung von genau 102 Bildpunkten pro Zeile. Jede Page besteht zudem noch aus jeweils 8 Spalten (D0 … D7), woraus sich eine Auflösung von 64 Bildpunkten pro Spalte ergibt.

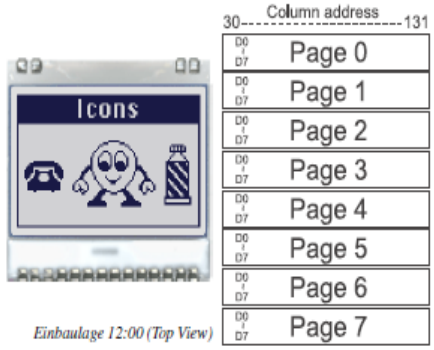


Abbildung 18: Einteilung des Bildschirms in "Pages" (Top View)

Die allgemeine Pinbelegung und die Beschaltung des Displays im Projekt sind in Abbildung 19 und Tabelle 21 auf der nachfolgenden Seite dargestellt:

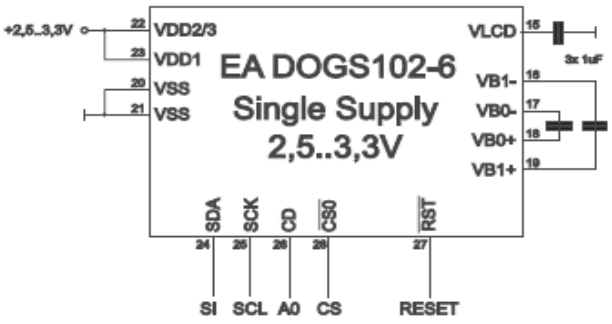


Abbildung 19: Allgemeine Pinbelegung des Displays

Bezeichnung	Pin	Beschaltung im Projekt
VLCD	15	GND
VB1-	16	Über Kondensator mit VB1+
VB0-	17	Über Kondensator mit VB0+
VB0+	18	Über Kondensator mit VB0-
VB1+	19	Über Kondensator mit VB1-
VSS	20/21	GND
VDD2/3	22	VCC
VDD1	23	VCC
SDA/SI	24	MOSI
SCK/SCL	25	SCK
CD/A0	26	PD5
$\overline{RST}$/RESET	27	PD4
$\overline{CS0}$/CS	28	PD7

Tabelle 21: Beschaltung des Displays, Pinbelegung vgl. [17], S. 2

Das *EA DOGS102-6* ist für eine unidirektionale Übertragung ausgelegt. Das heißt, dass es in einer Schaltung nicht als Master, sondern nur als Slave eingesetzt werden kann. Daraus folgt ebenfalls, dass Daten von der SPI-Schnittstelle, über die das Display mit dem Mikrocontroller verbunden ist, lesbar, aber nicht schreibbar sind.

Die Übertragung kann mit einer Taktrate (SCL) von bis zu 33 MHz erfolgen und folgt dem SPI-Mode 3 (Das höchstwertigste Bit wird zuerst gesendet). Dies muss unbedingt mit dem Modus der anderen Teilnehmer an der SPI- Leitung abgestimmt werden, da es sonst zu Übertragungsfehlern kommt.

Bevor das Display eingesetzt werden kann, ist eine Reihe von softwareseitigen Initialisierungen erforderlich, die zu Beginn des Programmablaufs erfolgen und in der Funktion *init_Display()* enthalten sind. Ausführliche Erläuterungen dessen finden sich im Programmcode selbst und sollen hier nicht weiter ausgeführt werden, da bei den Initialisierungsfunktionen und Grundausgabefunktionen des Displays und des Touchsensors auf bereits vorhandene Programmbausteine zurückgegriffen wurde.

Ob über die Datenleitung Kommando- oder Datenbits an den Bildschirm übertragen werden, wird über das Bit A0 (Pin 26) festgelegt. Ist es gesetzt, erfolgt eine Datenübertragung, wird es gelöscht, handelt es sich um ein Kommando (Beispielsweise eine Kontrastveränderung am Display). Daraufhin werden die Daten- oder Kommandobits gesendet. Nach jeder abgeschlossenen Übertragung muss die Verbindung von Master und Slave wieder über das Hochsetzen der Chip-Select-Leitung getrennt werden.

4.5.4 Funktionen zur grafische Darstellung auf dem Display

Bei den folgenden Abbildungen handelt es sich (außer es ist explizit im Beschreibungstext angegeben) um selbst einprogrammierte Muster, die in der Datei „display_blutzucker.c“ auf der beigefügten CD zu finden sind. Der zugehörige Quellcode befindet sich unter der jeweiligen Bildbeschreibung, sofern es sich um eine eigen erstellte Implementierung handelt. Die Grafikelemente selbst wurden in hexadezimalen Bitwerten eingegeben und befinden sich in der Datei "display_blutzucker_graphics.dat“.

Das Messgerät wird über eine Berührung auf dem Touchfeld gestartet. Es folgt die Darstellung des WieDAS-Logos für einige Sekunden. Hierbei handelt es sich um einen vorgefertigten Programmbaustein („void draw_LOGO (char* devicename)“ der für dieses Projekt aus Startbild zur Verfügung gestellt wurde.

Es erscheint auf dem Display die Anweisung, nun den Messstreifen einzulegen. Dargestellt wird dies durch die Darstellung eines Teststreifens und einen OK-Button, welche durch die folgenden Funktionen implementiert wird:

Die Funktion zum Zeichnen des Teststreifens ist:

```
void draw_MEASURINGTAPE (unsigned char locx, unsigned char locy)
{
                for (pagecounter = 0; pagecounter <=2; pagecounter++)
        {
                display_data (pagecounter+locy,locx,
pgm_read_byte(&pgm_measuringtape[pagecounter][0]));
                for (columncounter = 1; columncounter <=42; columncounter++)
                        display_data_ex
(pgm_read_byte(&pgm_measuringtape[pagecounter][columncounter]));
        }

}
```

Für den OK- Knopf war bereits eine Funktion vorhanden, die wiederverwendet werden konnte. Nach Berühren des Bildschirms werden die Koordinaten des Berührpunktes ausgewertet und wenn sie innerhalb der bestimmten Grenzen des Knopfes waren, wird die Messung gestartet.

Mit den Übergabewerten „locx“ und „locy“ kann die jeweilige Position der Grafik bestimmt werden. Diese Variablennamen werden auch in anderen Funktionen noch genutzt und definieren jeweils den Startpunkt der Darstellung.

Die Höhe des Bildes beträgt zwei Seiten und die Breite ist 42 Zeilen. Dies ist eine Festlegung, die bei der grafischen Entwicklung der Benutzeroberfläche getroffen wurde. Damit nicht die Werte des gesamten Bildschirms eingegeben werden müssen, werden lediglich die betroffenen Pages bzw. Zeilen mit Daten überschrieben. Dies geschieht in den „for“-Schleifen im Programmcode.

Ist das Einlegen des Messstreifens bestätigt worden, wird auf die Eingabe des Blutes auf den Sensor gewartet. Damit die Bedienung besonders einfach ist, wird dies durch eine Animation dargestellt, in der ein Tropfen Blut auf das markierte Sensorfeld zubewegt wird. Die Funktion „show_WAITING(…)“ (aufgerufen aus der Funktion test_messstreifen(…)) mit den Übergabeparametern für die Position des Tropfens und einem Bewegungsflag „move“ implementiert ebenfalls die Funktion „draw_MEASURIGTAPE(0,locy)“ und eine Funktion „draw_DROP(…)“, welche im folgenden Ablauf des Programms mehrfach über eine Schleife aufgerufen wird und den Tropfen so durch Veränderung der Variable „locx“ horizontal über den Bildschirm auf den Messstreifen zubewegt.

```
void show_WAITING(unsigned char locx, unsigned char locy, char move){
        draw_MEASURINGTAPE(0, locy);
        draw_DROP(locx, locy);

        if (move==1){                  //Animation Tropfen

                for (counter=0; counter >-35; counter--){
                        draw_DROP(locx+counter, locy);

                        _delay_ms(50);
                        clear_DROP(locx+counter+10, locy);
                        if (counter==-34){
                                clear_DROP(locx+counter, locy);

                        }

                }
        }


}


void draw_DROP (unsigned char locx, unsigned char locy)
{
                for (pagecounter = 0; pagecounter <=2; pagecounter++)
        {
                display_data (pagecounter+locy,locx,
pgm_read_byte(&pgm_drop[pagecounter][0]));
                for (columncounter = 1; columncounter <=10; columncounter++)
                        display_data_ex
(pgm_read_byte(&pgm_drop[pagecounter][columncounter]));

        }
}
```

Wurde eine Spannungsänderung über den AD-Wandler am Sensor festgestellt, wird die Messung des Blutzuckers gestartet. Dabei wird wieder auf die Funktion „draw_DROP(…)“ und einen Blinkmechanismus (abwechselndes Zeichnen und Löschen des Tropfens) zurückgegriffen. Die Funktionen „draw_DROP(…)“ und „clear_DROP(…)“ werden aus dem Programmcode „show_PROGRESS(…)“ in der Datei Blutzucker.c aufgerufen. Dies ist nötig, um möglichst geringe Verzögerungszeiten durch den Aufruf und die Ausführung des Blinkmechanismus zu erreichen. Außerdem wird dem Anwender der Fortschritt des Messvorgangs über einen Ladebalken, implementiert in den Funktionen „draw_LOAD(…)“ und „draw_LOADINGSEGMENT(…)“, veranschaulicht.

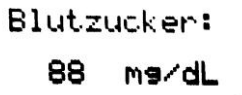
Der Umriss des Ladebalkens wird über die folgende Funktion dargestellt:

```
void draw_LOAD(unsigned char locx, unsigned char locy)
{
                for (pagecounter = 0; pagecounter <1; pagecounter++)
        {
                display_data (pagecounter+locy,locx,
pgm_read_byte(&pgm_load[pagecounter][0]));
                for (columncounter = 1; columncounter <=101; columncounter++)
                        display_data_ex
(pgm_read_byte(&pgm_load[pagecounter][columncounter]));

        }
}
```

Der Fortschritt der Messung wird mit der Funktion „draw_LOADINGSEGMENT(…)“ dargstellt. Zusätzlich zu den Koordinaten wird bei Aufruf dieser Funktion der Fortschritt über die Variable „progress“ übergeben. Dieser wird zur x-Koordinate addiert, wodurch immer ein weiterer Abschnitt rechts an den vorherigen angefügt wird und so den Fortschrittsbalken auffüllt.

```
void draw_LOADINGSEGMENT(unsigned char locx, unsigned char locy, unsigned char progress){
        display_data(locy,locx+progress,eeprom_read_byte(&ee_loadingsegment[0]));
}
```



Der Blutzuckerwert wird nach der Messung über eine einfache Textausgabe dargestellt.

4.6 Contiki OS

4.6.1 Allgemeines zum Betriebssystem

Bei Contiki handelt es sich um ein Open-Source Betriebssystem, welches IPv4 und IPv6 unterstützt und vornehmlich für den Einsatz in 8-Bit-Mikrocontrollern eingesetzt wird. Es zeichnet sich vor allem durch seinen sehr geringen Speicherverbrauch von nur wenigen Kilobytes und seine eventbasierte Struktur aus.

Contiki beinhaltet sogenannte „Protothreads“, die, da sie keinen eigenen Stack benötigen, um einiges weniger Speicherplatz benötigen als Threads und dennoch zur Parallelausführung gut geeignet sind. Beim Wechsel von Protothreads wurde auf Kosten der Variableninhalte ein hohes Maß an Dynamik erreicht. Die Inhalte sämtlicher, nicht als global oder statisch deklarierter Variablen werden bei einem solchen Wechsel gelöscht.

Da Contiki als eigenständiges Betriebssystem umfangreich und mitunter sehr komplex ist, wird an dieser Stelle auf weiterführende Literatur verwiesen. Einen guten Einstieg bietet da die im Literaturverzeichnis angezeigte Bachelorarbeit von Bernhard Esders mit dem Titel:

„Entwicklung, Implementierung und Evaluierung einer speichereffizienten, drahtlosen Kommunikation von eingebetteten Systemen mithilfe des IP-Protokolls basierend auf einem Multitasking-Betriebssystem für 8-Bit AVR Mikrocontroller.“

4.6.2 Projektbezogene Befehls- und Funktionsübersicht

Für die Datenübertragung der Blutzuckermesswerte über Funk wurde als Transportprotokoll UDP („User Datagram Protocol“) gewählt. Dabei handelt es sich um ein sehr schnelles, verbindungsloses Protokoll, welches im Gegensatz zu TCP („Transmission Control Protocol“) die korrekte Datenübertragung nicht sicherstellt.

Da zu diesem Zeitpunkt der Entwicklung noch nicht bekannt ist, an welche Teilnehmer die Messdaten via UDP gesendet werden sollen, entfällt die Konfiguration des Datenempfangspunktes dadurch, dass die Messdaten als Broadcast, d.h. an alle erreichbaren (bzw. verfügbaren) Endgeräte gesendet werden.

Grundsätzlich ist der Aufbau des Programms an die Struktur in Contiki derart anzupassen, dass die Hauptfunktion nicht mehr als „main“ bezeichnet werden darf. Contiki arbeitet mit Protothreads und die Hauptfunktion muss auch als solche deklariert werden. Dazu muss zunächst die Programmzeile

```
PROCESS(process name, char *strname);
```

bzw.im Projekt

```
PROCESS (bloodsugar, "bloodsugar");
```

ausgeführt werden. Damit wird der Prozess mit einem Prozessnamen definiert. Die „Main"-Funktion wird über die Zeile

```
PROCESS_THREAD(process name, process_event_t ev, void *data)
```

bzw. im Projekt

```
PROCESS_THREAD(bloodsugar, ev, data)
```

eingeleitet. Dabei muss der Name des Threads, sowie die Events, über die der Prozess angesprochen wird und die eventuell zu übergebenden Daten bei Prozessstart angegeben werden. Alle Befehle, die vor der Zeile

```
PROCESS_BEGIN()
```

definiert sind, werden bei jedem Kontextwechsel erneut ausgeführt. Diese Befehlszeile ist zwingend notwendig, genau wie das Makro

```
PROCESS_END()
```

welcher den Prozess bei der Ausführung dessen beendet. Mit dem Befehl

```
AUTOSTART_PROCESSES(&bloodsugar);
```

wird der Prozess, der beim Start des Programms automatisch ausgeführt werden soll, definiert.

Damit eine Übertragung über UDP mit Contiki funktioniert, muss als erstes über

```
struct uip_udp_conn *c
```

eine Referenz auf eine Verbindung über UDP angelegt werden. Darin werden dann anschließend Quell- und Zielport oder je nachdem andere relevante Daten, wie die Zieladresse, wenn nicht über Broadcast gesendet wird, angegeben.

Da hier ein UDP- Broadcast gesendet wird, muss lediglich der Port 1234 angegeben werden, weitere Parameter werden nicht benötigt, weshalb der zweite Übergabewert ein Null-Parameter ist. Damit eine Umwandlung der Portnummer von der Hostbyteorder in die Networkbyteorder erfolgt, wird das Makro HTONS() („Host to Network- Umwandlung") eingesetzt. Diese Konvertierung ist nötig, da es sich um ein plattformübergreifendes Netzwerkprotokoll handelt, bei der die Bytereihenfolge (Networkbyteorder) vorgeschrieben ist und sich von der Bytereihenfolge des Systems unterscheidet (Hostbyteorder).

```
udp_broadcast_new(u16_t port, void* appstate)
```

Bezogen auf das Projekt ergibt sich damit:

```
udp_broadcast_new(HTONS(4321),NULL);
```

Die Verbindung ist nun eingerichtet und bereit um Daten zu Versenden. Dazu ist es nötig, dem Stack über die Programmzeile

```
tcpip_poll_udp(struct uip_udp_conn *conn);
```

mitzuteilen, dass eine zu bearbeitende Verbindung vorliegt. Über das Makro

```
PROCESS_WAIT_EVENT_UNTIL(ev == tcpip_event);
```

wird gewartet, bis diese bereit ist. Daraufhin kann man über den einfachen Befehl

```
uip_send(const void *data, int len);
```

ein Datenpaket versenden. Dazu wird ein Zeiger auf die zu sendenden Daten und ein Parameter, der die Länge dieser Daten angibt, erwartet.

Als etwas problematisch bei der Nutzung von Contiki in Zusammenhang mit der vorliegenden Schaltung stellte sich heraus, dass die UART-Schnittstelle von diesem Betriebssystem standardmäßig eingeschaltet ist. Da über diese Ports die Displaybeleuchtung gesteuert wird, war es nötig, diese Funktion abzuschalten, da sie auch nicht benötigt wird. Dies wurde über das Löschen der entsprechenden Registerinhalte über die Befehlszeilen

```
UCSR1B &=0x00;
UCSR1C &=0x00;
```

durchgeführt. Dabei wurde jedoch bei der Erstausführung die Übertragung des Messwertes über UDP gestört, weshalb vor Abschalten des UART eine leere Dummy-Senderoutine implementiert werden musste. Die Übertragung war somit wieder von Beginn an möglich.

4.7 Weitere Bauteile und Funktionen

4.7.1 Erzeugung der Messspannung

Die Messung des Blutzuckerwertes benötigt eine konstante Spannung über den Messkontakten des Sensors. Um dies sicherzustellen, wird die angelegte Betriebsspannung (3,3V DC) zuerst mit Hilfe eines Spannungsstabilisierers auf konstante 2,5V DC gebracht. Dies ist nötig, da die Ausgangsspannung der einzusetzenden Batterien mit der Zeit leicht sinkt und so in diesem Fall nicht als konstant angesehen werden kann. Diese Spannung dient als Eingangsgröße für einen Operationsverstärker in Subtrahiererschaltung, wie in Abbildung 20 dargestellt. Schaltungen dieser Art zeichnen sich durch ihre hohe Spannungsstabilität am Ausgang aus und eignen sich somit sehr gut für den dargelegten Fall. Ohne den Widerstand R7 und mit U_{e2} verbunden mit Masse läge ein idealer invertierender Verstärker vor. Das würde bedeuten, dass die Ausgangsspannung U_a bei $U_{e1} = U_{e2}$ genau 0V ist. Im Realfall ist es nötig, die Eingangsruheströme und den damit einhergehenden Offsetfehler zu kompensieren, dafür wird der Widerstand R7 eingefügt. Damit ergeben sich mit Hilfe des Überlagerungsverfahrens folgende Gleichungen:

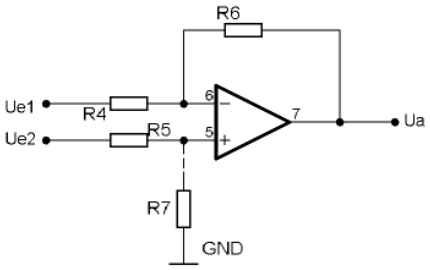


Abbildung 20: OP in Subtrahiererschaltung

Als verstärkte Teilspannung durch Nullsetzen von U_{e2}:

$$U_a' = -\frac{R6}{R4} \cdot U_{e1}$$

Als verstärkte Teilspannung durch Nullsetzen von U_{e1}:

$$U_a'' = \frac{R7}{R5 + R7} \cdot (1 + \frac{R6}{R4}) \cdot U_{e2}$$

Werden nun beiden Teilspannungen miteinander addiert, erhält man die Ausgangsspannung U_a:

$$U_a = \frac{R7}{R5 + R7} \cdot (1 + \frac{R6}{R4}) \cdot U_{e2} - \frac{R6}{R4} \cdot U_{e1}$$

Die Eingangsspannungspannungen sind gleich der Ausgangsspannung des Spannungsstabilisierers $U_{e2} = U_{e1}$ = 2,5V. Zur Messung des Blutzuckerwertes benötigt man eine Messspannung von 400…600mV. Mit den in der Schaltung eingesetzten Widerständen R4=R6=R7=10kΩ und R5=R5.1+R5.2=6,9kΩ ergibt sich eine Ausgangs- bzw. Messspannung von $U_a \approx 460$mV. In der Praxis liegt dieser Wert etwas höher, aber durchaus noch im tolerierbaren Bereich. Da dieser Spannungswert konstant ist, kann er für die folgenden Messungen gut verwendet werden.

Die Operationsverstärker- Versorgungsspannung liegt bei +3,3V und Masse. Da beide Grenzwerte nicht erreicht werden, ist es nicht unbedingt nötig, einen Rail- to- Rail- OP einzusetzen. Im Hinblick auf weitere AAL- Projekte und die mögliche Kombinierung des Blutzuckermessgerätes mit etwaigen anderen Messgeräten (z.B. Blutdruck) wurde der Operationsverstärker *LM324D* gewählt, da in diesem Baustein 4 einzelne OP's integriert sind.

4.7.2 Messstreifenerkennung

Am Anfang des Projektes wurde das Blutzuckermessgerät *SQL25* von *Sanitas* mit den dazugehörenden Messstreifen eingesetzt. Diese zeichneten sich vor allem durch ihren einfachen Aufbau aus, weshalb sie einen guten Einstieg in das Verständnis der Blutglukosemessung gaben. Bei diesem Messstreifen waren nur drei Kontakte vorgesehen:

- Der Pin für die Messspannung (Abbildung 21, linker Pin)
- Der Massepin (Abbildung 21, rechter Pin)
- Und der über den Messstreifen mit dem Massepin kurzgeschlossene Sensorkontrollpin (Abbildung 21, mittlerer Pin)

Abbildung 21: Schemazeichnung des Teststreifens von "Sanitas"

Dementsprechend wurde die Kontrolle, ob ein Teststreifen eingelegt wurde, über den Zustand des Mikrocontrollereingangs PD2 abgefragt. Dieser wurde in den High- Zustand versetzt, wenn der Messstreifen eingelegt wurde und so eine Spannung über diesem auftrat. Da der Kontrollpin mit Masse verbunden ist, wird das Messergebnis dadurch nicht verfälscht.

Da man jedoch für dieses Messgerät keine Teststreifen mehr bekommen konnte, war es nötig, das Projekt mit einem anderen Referenzgerät, dem *Contour* von der Firma *Bayer*, bzw. dessen Blutzuckersensor weiterzuführen. Dieses Messgerät ist weitaus genauer und eines der führenden auf dem Markt. Hier findet nicht nur eine Teststreifenerkennung und eine Blutzuckermessung, sondern auch eine Messung der Blutmenge statt. Dadurch ist das Gerät wesentlich genauer und zuverlässiger. Jedoch ist der Teststreifen wesentlich komplizierter aufgebaut und das vorher angewandte Prinzip der Teststreifenerkennung war nicht mehr einsetzbar. Zwar konnten minimale Spannungsspitzen von ca. 5mV am Sensorkontrollport gemessen werden, wenn ein Teststreifen eingelegt wurde, diese waren jedoch stark alternierend in der Amplitude und in ihrem Auftreten, so dass davon ausgegangen werden musste, dass die Erkennung eines Messstreifens im Sensor auf eine andere Weise funktionierte und es sich bei den gemessenen Spannungswerten nur um Zufallswerte bzw. Rauschen handelte.

Um die Möglichkeit einer derartigen Kontrolle jedoch zu prüfen, ohne den Sensor durch Berührung oder andere äußere Einflüsse zu beeinträchtigen, wurde die Schaltung um einen Elektrometerverstärker

erweitert, der an den Kontrollpin des Sensors angeschlossen war und aus der minimalen Signalamplitude bei Einlegen des Streifens einen Wert, der den Zustand des Einganges PD2 am Mikrocontroller vom Low- in den High- Zustand ändern konnte. Es stellte sich jedoch heraus, dass eine Lösung wie diese nicht realisierbar war und keine brauchbaren Verhaltensweisen der Messschaltung lieferte.

Da eine Erkennung des Messstreifens jedoch von essentieller Wichtigkeit ist, wurde im Projekt eine Steuerung über den Touchscreen eingefügt. Hier kann zwar nicht automatisch das Einlegen überprüft werden, jedoch muss der Nutzer des Messgerätes manuell über Berühren des Bildschirms bestätigen, dass ein Teststreifen eingelegt wurde.

4.7.3 Auswertung des Messwertes

Mit Hilfe des bereits erwähnten Referenzgerätes von *Sanitas* wurde als Basis für das Projekt als erstes analysiert, wie eine Blutzuckermessung überhaupt stattfindet, denn dazu gab es sehr wenige aussagekräftige Quellen. Man konnte jedoch die Möglichkeiten derart reduzieren, als dass es sich um eine sog. „Amperometrische Blutzuckermessung“ handeln musste. Hierbei wird der Blutglukosewert über den Strom, der über den Messstreifen fließt und vom Zuckergehalt des Blutes abhängig ist, gemessen. Im Anschluss daran wurden zahlreiche Messungen mit dem Oszilloskop am Sensor zwischen den Messklemmen getätigt. Auffällig war dabei, die bei verschiedenen Blutzuckerwerten ähnlich aussehende Kurve, die einer e- Funktion (siehe Abbildung 22) zu folgen schien. Im Referenzgerät wurde der Sensor mit einer Messspannung von etwa 400mV beaufschlagt. Damit der Strom, bzw. der Spannungsabfall über einem Messwiderstand gemessen werden konnte, musste der Sensor um einen solchen erweitert werden. Da nicht bekannt war, in welchen Dimensionen sich der Widerstandswert von Blut befindet, wurde ein Messwiderstand von 1kΩ eingesetzt. Dieser sollte die Messung durch seinen Eigenwiderstand so wenig wie möglich beeinflussen und so die Ergebnisse verfälschen. In Abbildung 23 ist eine Messkurve, wie eben erwähnt, dargestellt. Nach einigen Untersuchungen der Kurven wurde festgestellt, dass diese sich nicht in der

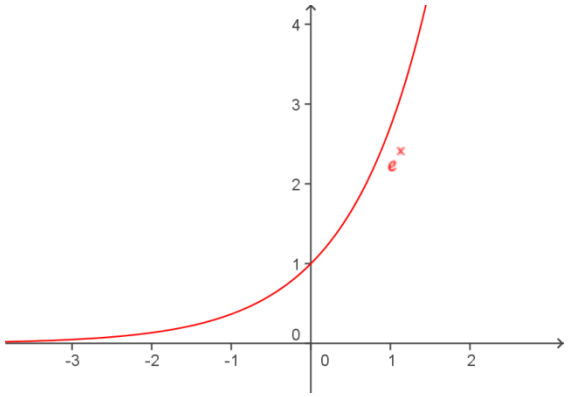


Abbildung 22: Exponentialfunktion

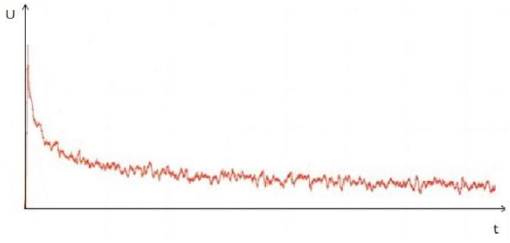


Abbildung 23: Beispiel für eine Blutzucker-Messkurve mit dem Blutglukosewert 106mg/dl; Y: Spannung über dem Messwiderstand; X: Zeit

allgemeinen Form, sondern in der Höhe der Amplitude bei Start der Messung und Geschwindigkeit des Spannungsabfalls mit der Zeit änderte. Dies ließ die Vermutung zu, dass es sich um eine Auswertung der entstandenen e-Funktion mit Hilfe einer Integration handeln könnte und somit eine Flächenberechnung unter der Kurve wäre. Dazu wurden bei den einzelnen Kurven zuerst per Hand die Zeitkonstante τ bestimmt. Es stellte sich jedoch heraus, dass diese Methode viel zu ungenau war und auch mit Hilfe eines Mikrocontrollers, der die Zeitkonstanten jeweils berechnen könnte, viel zu lange dauern würde. Deshalb wurde eine andere Möglichkeit der Berechnung des Blutzuckerwertes gesucht.

Integrieren bedeutet nichts anderes als Summenbildung der einzelnen Flächen unter einer Kurve zu jedem Zeitpunkt. Wie in Abbildung 24 dargestellt, wird die Flächenberechnung der Funktion umso genauer, je kleiner die Zeitintervalle bei der Signalabtastung gewählt werden.

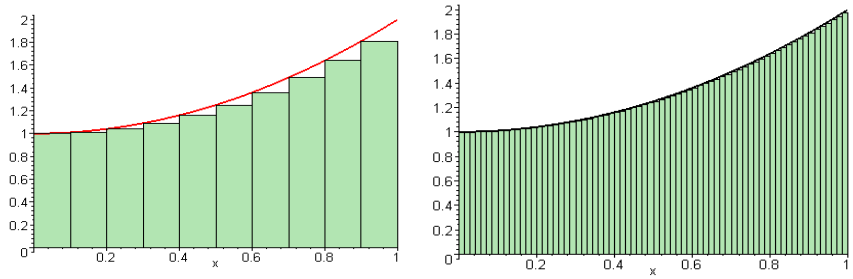


Abbildung 24: Integrieren heißt Summenbildung; links: grobe Zeitintervalle, rechts: feine Zeitintervalle

Dieses Zeitintervall T wurde im Projekt auf 50ms festgelegt. Bei den 100 Signalabtastungen pro Messung ergibt das eine Messzeit von etwa 5s. Da der Wert der Messkurve jedoch stark schwankt und von einem Rauschen überlagert ist, wird alle 50ms nicht nur eine einzelne Messung gemacht, sondern mehrere direkt nacheinander und davon der Mittelwert gebildet, der in der Variable *teilWert* gespeichert wird. Dieser wird, um die Möglichkeit zu liefern, die einzelnen Messwerte auszugeben, im Array *messWerte[counter]* gespeichert und zuletzt auf die Werte der vorherigen Abtastungen aufsummiert. Diese Summe wird durch die Anzahl der Messungen (hier: 100) geteilt, der so entstandene Mittelwert in der Variablen *ergebnisSumme* zwischengespeichert und repräsentiert nun nach abgeschlossenem Messvorgang den Blutzuckerwert als Ausgabewert des AD-Wandlers. Dieser muss nun noch in den Blutzuckerwert umgewandelt werden. Damit dies möglich ist, wurden das Referenz- und das bis dato entwickelte Blutzuckermessgerät mit verschiedenen Widerstandswerten ohne eingebauten Sensor an den Messklemmen ausgemessen. Aus diesen Messungen wurde dann:

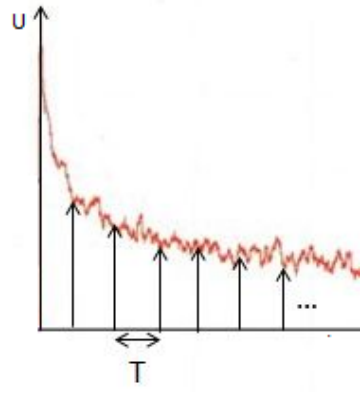


Abbildung 25: Abtasten des Messwertes

- Der Widerstandswert des Messwiderstandes (hier: 68kΩ) am Sensor bezogen auf
- den Blutzuckermesswert des Originalgerätes
- den AD-Wert des entwickelten Gerätes

und so eine Formel für die Berechnung des Blutglukosewertes aus den Ausgabewerten des AD-Wandlers erstellt. Dazu wurden zunächst die Messwerte als Basis für ein Diagramm genommen und eine Ausgleichsgerade durch die Messpunkte gelegt via *Microsoft Excel* (siehe Abbildung 26).

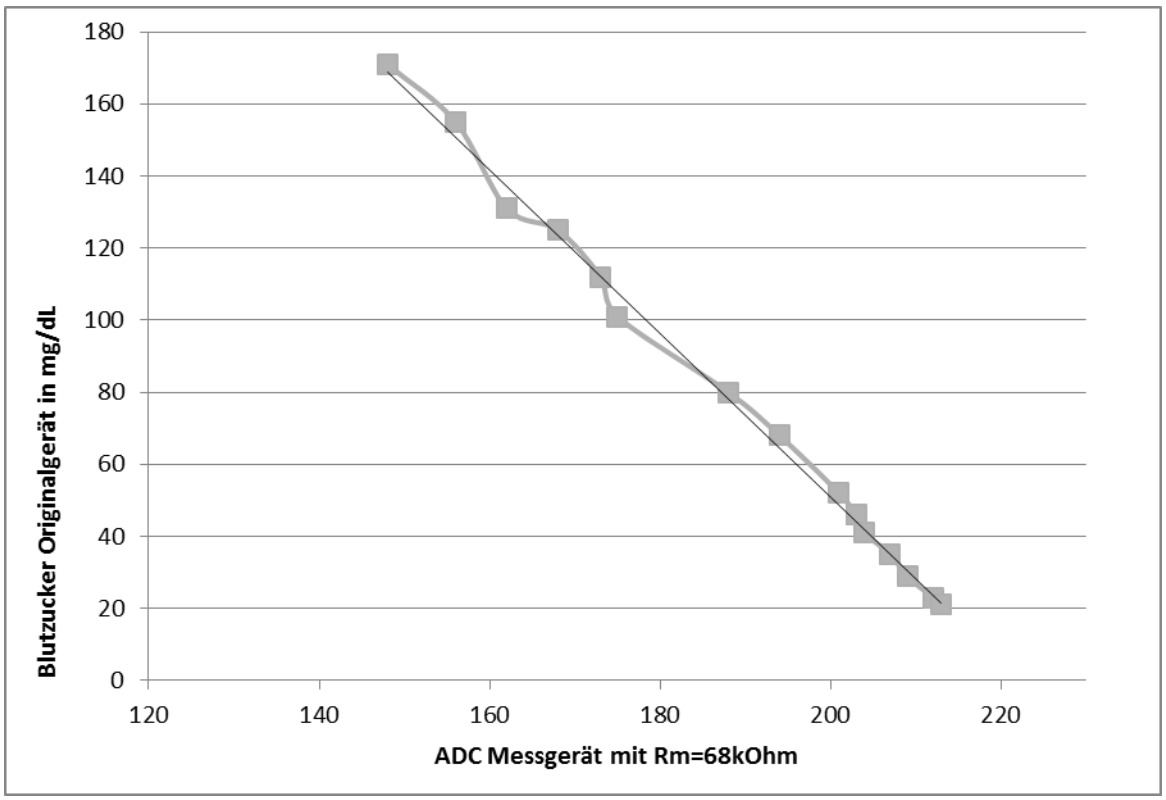


Abbildung 26: Abhängigkeit des Blutzuckerwertes vom ADC-Wert

Mit Hilfe dieses Programms konnte die Formel zur Berechnung der Ausgleichsgeraden direkt angegeben werden. Diese lautet auf das Projekt bezogen:

$$\frac{Blutzuckerwert}{mg/_{dl}} = -2{,}266 \cdot ADC + 504{,}28$$

Da im Projekt jedoch die Sensoren ausgetauscht wurden, musste die Formel angepasst werden. Der Sensor von *Sanitas* lieferte ohne Belastung einen ADC-Wert von 220. Darauf waren die Referenzwerte des Originalgerätes ausgelegt. Der Bayer-Sensor lieferte jedoch einen ADC-Wert von 254-255 im unbelasteten Zustand. Einige Versuche haben gezeigt, dass die Differenz der beiden Werte vom Mittelwert für die Berechnung des Blutzuckers abgezogen werden muss. Damit ergibt sich die Berechnungsformel:

$$\frac{Blutzuckerwert}{mg/_{dl}} = -2{,}266 \cdot (ADC - 35) + 504{,}28$$

In Tabelle 22 sind die Abhängigkeiten von Widerstands-, Blutzucker- und ADC-Wert, sowie die berechneten Werte des Blutzuckers mit dem im Rahmen dieses Projektes entwickelten Messgerätes und deren Abweichung zum Referenzwert dargestellt.

Widerstand in kΩ	Blutzuckerwert Originalgerät in mg/dL	Gemessener ADC-Wert	Berechneter Blutzuckerwert	Abweichung in %
47	358	128	214,232	-40
98	171	148	168,912	-1,2
108	155	156	150,784	-2,7
124	131	162	137,188	4,7
130	125	168	123,592	-1,1
145	112	173	112,262	1,2
159	101	174	107,730	6,7
197	80	188	78,272	-2,2
220	68	194	64,676	-4,9
252	52	201	48,814	-6,1
265	46	203	44,282	-3,7
279	41	204	42,016	2,5
298	35	207	35,218	0,6
320	29	209	30,686	5,8
346	23	212	23,888	3,9
353	21	213	21,622	3.0
470	-	219	8,026	-

Tabelle 22: Zusammenhang zwischen Widerstands-, ADC- und Blutzuckermesswert

Auffällig ist vor allem die Abweichung des Referenzwertes zum berechneten Wert bei einem Soll-Blutzucker von 358 mg/dl. Diese Abweichung kommt zustande, weil der ADC schaltungsbedingt keine kleineren Werte als 128 abgeben kann, denn dies ist der Messwert des in den Sensor eingelegten Messstreifens selbst. Der Maximalwert, der ausgegeben werden kann, hat also den ADC-Wert 129 und ist umgerechnet ein Blutglukosewert von 212 mg/dl. Aufgrund der Messbereichsgrenze des *Sanitas*-Gerätes ist es nicht möglich, Werte unter 20 mg/dl zu messen, da dies unter dem dort darstellbaren Wert liegt. Somit ist der Messbereich des entwickelten Blutzuckermessgerätes 20 … 212 mg/dl mit einer maximalen Abweichung von etwa 7% zum Originalgerät, welches selbst laut Hersteller eine Genauigkeit von ±20% hat. Dies kann im Ernstfall schlimme Folgen haben und ist deshalb nicht zu tolerieren. Eine erneute Referenzmessung mit einem exakteren Messgerät muss vor dem Einsatz des hier vorgestellten Gerätes erfolgen und dieses über die Berechnungsformel entsprechend angepasst werden.

Der programmtechnische Ablauf der Messung ist der folgende:

Durch Contiki wird bei Programmstart der Prozess *bloodsugar* automatisch gestartet:

```
//Prozess:bloodsugar
PROCESS (bloodsugar, "bloodsugar");

//Thread: test
PROCESS_THREAD(bloodsugar, ev, data)
{
        PROCESS_BEGIN();

        c=udp_broadcast_new(HTONS(4321),NULL);
        etimer_set(&timer, CLOCK_SECOND*.5);
        InitAll();
        SLEEP_INIT();
        etimer_set(&timer, CLOCK_SECOND*0.5);
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&timer));

//Damit die Senderoutine funktioniert muss zu Anfang einmal eine Dummy-Übertragung
//erfolgen
        tcpip_poll_udp(c);
        PROCESS_WAIT_EVENT_UNTIL(ev==tcpip_event);
        uip_send("",0);
        etimer_reset(&timer);
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&timer));

//Ausschalten der seriellen Schnittstelle, sonst kann die Displaybeleuchtung wegen
//Doppelbelegung von PD2 nicht abgeschaltet werden
        UCSR1B &=0x00;
        UCSR1C &=0x00;

        reset_TOUCH(0);
        touch_x=-1;
        touch_y=-1;

        while (sleep==0)
        {
//Wenn Schlafmodus nicht aktiv->Programmstart

                reset_TOUCH(0);
                adcWert=0;
                mittelwert=0;
                progresscount=0;

                while(touch_x==-1 && touch_y ==-1)
                {
//Warten, bis der Bildschirm berührt wird
                        touch_x = get_TOUCH('x');
                        touch_y = get_TOUCH('y');
                        reset_TOUCH(1);
                }

        LED_DISPLAY(1);                                 //Displaybeleuchtung einschalten
        display_clear();

        draw_LOGO("  Blutzucker  ");                    //Anzeigen des Logos für 5s
        _delay_ms(5000);
        display_clear();

        test_messstreifen();
//Aufruf der Funktion zum Erkennen des Messstreifens und des Blutes auf diesem

        messung();                                      //Messfunktion

        etimer_set(&timer, CLOCK_SECOND*0.5);
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&timer))
        tcpip_poll_udp(c);
        PROCESS_WAIT_EVENT_UNTIL(ev==tcpip_event);
        uip_send(sendstr,strlen(sendstr));              //senden des BZ-Wertes
        etimer_reset(&timer);
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&timer));
```

```
//Ausschalten der seriellen Schnittstelle, sonst kann die Displaybeleuchtung wegen
//Doppelbelegung von PD2 nicht abgeschaltet werden
        UCSR1B &=0x00;
        UCSR1C &=0x00;

        display_clear();
        LED_DISPLAY(0);                                 //Displaybeleuchtung ausschalten
        reset_TOUCH(0);

        touch_x=-1;
        touch_y=-1;

        sleep =1;
        SLEEP_ALLOW();
//Display ausgeschaltet lassen und Microcontroller in Schlafmodus versetzen, bis eine
//erneute Eingabe auf dem Touchscreen erfolgt

                while(sleep ==1)
                {
                        if((touch_x==-1) && (touch_y ==-1))
                        {
                        touch_x = get_TOUCH('x');
                                touch_y = get_TOUCH('y');
                                reset_TOUCH(1);
                        }
                        else
                        {
                                sleep =0;
                                SLEEP_DISALLOW();
                        }
                }

        }
        PROCESS_END();

}

AUTOSTART_PROCESSES(&bloodsugar);
//Prozess, der automatisch gestartet werden soll
```

Nachdem das Logo dargestellt wurde, wird zuerst über die Funktion test_messstreifen() auf die Bestätigung des Einlegens eines Messstreifens gewartet und anschließend eine Messung über den ADC durchgeführt. Über diese wird, sobald Blut auf den Sensor gegeben wurde und der ADC-Wert dadurch unter einen Schwellwert (hier: 245) sinkt, die Messung gestartet. Der hier dargestellte Quellcode enthält nur die im abgeschlossenen Projekt genutzte Funktion der Messstreifenerkennung via Bestätigungsknopf. Im Quellcode auf der beiliegenden CD befinden sich zudem die auskommentierten Programmzeilen, mit denen eine Erkennung des Messstreifens von *Sanitas* möglich war und funktionierte.

```
void test_messstreifen()
{

        show_TEXT(2,0,"Teststreifen",0,0);
        show_TEXT(5,1,"einlegen",0,0);
        draw_MEASURINGTAPE(0,4);
        draw_OK(70,4);

        reset_TOUCH(0);
        touch_x=-1;
        touch_y=-1;

        while
(!((touch_y > 20) & (touch_y < 60) & (touch_x > 4) & (touch_x < 30)))
        {
                touch_x = get_TOUCH('x');
                touch_y = get_TOUCH('y');
                reset_TOUCH(1);
        }
```

```
                        display_clear();
                        _delay_ms(200);
//Warten, damit keine Schwingungen in die folgende Messung eingehen

                        ADMUX = 0;
                        ADMUX &= ~(1<<REFS1);
                        ADMUX |= (1<<REFS0);
                        ADMUX |= (1<<MUX0);
                        ADMUX |= (1<<MUX3);
                        ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1);

//solange keine Änderung am Sensor auftritt soll die Messung nicht gestartet werden
                        do {
                                show_WAITING(80,2,1);
                                ADCSRA |= (1<<ADSC);
                                while (ADCSRA & (1<<ADSC));
//Warten auf Ende der Wandlung

                                adcWert = ADCW;
                                adcWert /=4;

                        } while ((adcWert>=245)&(adcWert<=255));
//254: Standardwert; 245: Schwellwert für die Messung

}
```

Zuerst wird das Signal wie eben beschrieben über die Funktion messung() abgetastet. Währenddessen erfolgt eine Anzeige des Fortschrittes der Messung über den Bildschirm mit der Funktion calc_PROGRESS()

```
void messung()
{

        ADMUX = 0;
        ADMUX &= ~(1<<REFS1);
        ADMUX |= (1<<REFS0);
        ADMUX |= (1<<MUX0);
        ADMUX |= (1<<MUX3);
        ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1);


        display_clear();
        show_TEXT(2,0,"...Messung...",0,0);
        draw_LOAD(0,6);

        ergebnisSumme=0;

        for (counter=0;counter <100; counter++)
        {

                ADCSRA |= (1<<ADSC);

                while (ADCSRA & (1<<ADSC));
                                                        //Warten auf Ende der Wandlung

                for (count2 =0; count2 <5; count2++)
                {

                        adcWert = ADCW;
                        adcWert /=4;

                        ADCSRA |= (1<<ADSC);
                        while (ADCSRA & (1<<ADSC));
                        teilWerte[count2]=adcWert;

                }

                teilWert= (
                (int16_t)teilWerte[0]+
                (int16_t)teilWerte[1]+
                (int16_t)teilWerte[2]+
                (int16_t)teilWerte[3]+
```

```
                (int16_t)teilWerte[4])/5;
                                                //Mittelwert der Teilwerte bilden

                messWerte[counter]= teilWert;
                ergebnisSumme =ergebnisSumme + teilWert;
                                        //aktuellen Messwert zurückgeben

                _delay_ms(50);

                calc_PROGRESS(counter);
                                                //Fortschritt berechnen & ausgeben

        }

        mittelwert = ergebnisSumme/100;
        display_clear();
        get_OUTPUT(mittelwert);

}


void calc_PROGRESS(int progress)
{
        draw_LOADINGSEGMENT(1,6,progress);

        if (progress ==1*progresscount)
        {
                draw_DROP(45,2);
        }
        else if (progress ==1*progresscount+5)
        {
                clear_DROP(45,2);
                progresscount =progresscount+10;
        }

}
```

Der Mittelwert der Messungen wird gebildet und an die Funktion getOutput() übergeben.

```
void get_OUTPUT(unsigned char mittelwert)
{

        if (mittelwert>128 && mittelwert <213)
                                                //Grenzen des Messbereichs
        {
                display_clear();
                calc_Blutzucker(mittelwert);
        }

        else if (mittelwert <=128)
        {
                show_TEXT(2, 2, "Blutzucker:", 0, 0);
                show_TEXT(2, 4, ">220", 0, 0);
                sendstr="Blutzucker: >220";
                                                        //für Ausgabe über Funk
                _delay_ms(5000);
                display_clear();
        }

        else if (mittelwert >=214 )
        {
                show_TEXT(2, 2, "Blutzucker:", 0, 0);
                show_TEXT(2, 4, "<20", 0, 0);
                sendstr="Blutzucker: <20";
                                                        //für Ausgabe über Funk
                _delay_ms(5000);
                display_clear();
        }
        else if (mittelwert >= 250)
```

```
        {

//Wird kein Blut eingegeben, entsteht der Messwert (ADC-Wert) >250
                show_TEXT(2, 2, "Fehler", 0, 0);
                show_TEXT(0, 4, "Neustart erfolgt", 0, 0);
                _delay_ms(5000);
                display_clear();
        }

        else
        {
                show_TEXT(2, 2, "Fehler", 0, 0);
                show_TEXT(0, 4, "Neustart erfolgt", 0, 0);
                _delay_ms(5000);
                display_clear();
        }
}
```

- Ist der Wert ≤128 wird als Messwert >220 mg/dl angegeben
- Ist der Wert≤214 wird als Messwert <20 mg/dl ausgegeben
- Liegt der Wert innerhalb des Messbereichs, wird dieser an die Funktion calc_Blutzucker() übergeben, der Wert berechnet und auf den Display sowie über Funk ausgegeben

```
void calc_Blutzucker(unsigned int mittelwert)
{
        char mwArray[10];

//Ausgabe des Blutzuckers ?ber das Array
        utoa (mittelwert,mwArray,10);

        double ergebnisBlutzucker=-2.266*(mittelwert-35)+504.28;
//Formel zur Berechnung des Blutzuckers ?ber den ADC-Wert; -35 Da ADC-Werte der
//Messstreifen unterschiedlich sind
        int ergBz = (int)ergebnisBlutzucker;

        char blutzuckerArray[10];
                                                //Speichern des Blutzuckers in das Array und
        itoa (ergBz,blutzuckerArray,10);        //Konvertierung von int zu ASCII

        show_TEXT(3, 2, "Blutzucker:", 0, 0);
        show_TEXT(5, 4, blutzuckerArray, 0, 0);
        show_TEXT(9, 4, "mg/dL", 0, 0);

        _delay_ms(5000);
        display_clear();

        char *Name= "Blutzucker/(mg/dL): ";
        strcpy(sendstr, Name);
        strcat(sendstr, blutzuckerArray);

        ergBz=0;
        mittelwert=0;
        progresscount=0;

}
```

5 Bedienung des Messgerätes

Wollen Sie ihren Blutzucker selbst messen, benötigen Sie:

- Das Blutzuckermessgerät,
- Einen Teststreifen,
- Eine Lanzette, bzw. Nadel und die
- Stechhilfe

Wichtig:

- **Achten Sie unbedingt darauf, dass Sie bei jeder Messung (auch wenn Sie mehrmals bei sich selbst den Blutglukosespiegel messen möchten) eine frische, unbenutzte Lanzette nehmen, da die Verwendung verschmutzter Nadeln zu bakteriellen Infektionen führen kann!**

- **Fassen Sie die Teststreifen nur mit trockenen Fingern an und verschließen Sie die Dose nach der Entnahme des Streifens sofort wieder.**

- **Die Teststreifen haben ein Verfallsdatum, welches unbedingt eingehalten werden sollte, da die Messergebnisse ansonsten verfälscht werden können.**

- **Beachten Sie, dass beim vorgestellten Messgerät der Blutzuckerwert in mg/dl angegeben wird (1 mg/dl = 1/18,02 mmol/l).**

Abbildung 27

Vorbereitung:

Waschen Sie sich mit warmem Wasser gründlich die Hände und trocknen Sie sie gut ab, damit durch das Wasser das Messergebnis nicht verfälscht wird. Außerdem wird dadurch die Durchblutung in den Fingern gefördert und die Blutentnahme erheblich vereinfacht. Eine Desinfizierung der Einstichstelle ist nicht notwendig.

Brechen sie die Verschlusskappe vorsichtig von der Nadellanzette ab.

Legen Sie die Lanzette in die Stechhilfe ein.

Stellen Sie die Tiefe des Nadeleinstichs entsprechend ein:

Einstichtiefe	Hauttyp
1-2	Dünne / weiche Haut
3	Normale Haut
4-5	Dicke / schwielige Haut

Tabelle 23: Richtwerte für die Einstellung der Stichtiefe

Ziehen Sie am hinteren Teil der Stechhilfe, bis ein leises Klicken zu hören ist.

Starten Sie das Messgerät durch Fingerdruck auf den Bildschirm.

Legen Sie einen Teststreifen in das Messgerät ein, bestätigen Sie mit Druck auf den dargestellten Knopf und warten Sie, bis Sie aufgefordert werden, Blut auf den Sensor zu geben.

Durchführung:

Setzen Sie die Stechhilfe mit der Spitze seitlich an die Fingerbeere an und drücken Sie auf den Rundknopf, der sich etwas oberhalb des Einstellrades für die Stichstärke befindet.

Geben Sie den Blutstropfen in das Kapillarfeld.

Die Messung wird gestartet und das Ergebnis wird auf dem Display angezeigt.

6 Optimierungsmöglichkeiten

Bei dem hier vorgestellten Blutzuckermessgerät handelt es sich um einen Prototypen, der die wichtigsten Grundfunktionen bereitstellt. Allerdings gibt es noch einige Punkte, an denen Verbesserungen vorgenommen werden können, auf die in diesem Kapitel näher eingegangen werden soll.

Zu allererst gab es Probleme bei der Messstreifenerkennung mit dem Sensor von *Bayer*. Zum einen kann hier eine automatische Erkennung des Teststreifens stattfinden. Dazu ist es jedoch nötig, den Sensor und die von *Bayer* eingesetzte Methode zur Messstreifenerkennung zu untersuchen und entsprechend umzusetzen. Da die Erkennung scheinbar nicht wie beim *Sanitas*-Gerät über einen Kurzschluss erfolgt, könnte es sein, dass der Sensor von *Bayer* eine mechanische Vorrichtung im Inneren besitzt, die das Einschieben des Teststreifens registriert und an den Mikrocontroller meldet. Messungen am Teststreifen haben jedoch auch ergeben, dass sich der Widerstand des Messstreifens beim Einschiebevorgang ändert. Die Kontakte dessen sind unterteilt, also wäre es ebenso möglich, dass eine Änderung des Widerstandes am Sensor eine Erkennung des Teststreifens auslöst.

Außerdem könnte das Messgerät dahingehend modifiziert werden, dass es nicht auf Berührung des Bildschirms, sondern bei Einstecken des Teststreifens gestartet wird. Da der Kontrollport bereits in der vorliegenden Schaltung an einem Interruptpin INT0 (PD2) angeschlossen ist, könnte man dies mit Hilfe einer Flankenauswertung und einem Interrupt auslösen.

Weiterhin erkennen die im Handel erhältlichen Blutglukosemessgeräte meist einen verdreckten, oder bereits benutzten Teststreifen und geben dementsprechend einen Fehler aus.

Da die Messung auf einem Blutzuckermessgerät beruht, dass selbst eine relativ große Ungenauigkeit besitzt, und die Messwerte hier noch einmal eine Messungenauigkeit durch die Linearisierung der Messkurve, Rundungsfehler und die Auflösung des AD-Wandlers entstehen, sollte auch die Genauigkeit des Messgerätes noch weiter verbessert werden. Dies könnte zum einen durch Verbesserungen an den vorher genannten Punkten erfolgen, aber es gibt auch weitere Möglichkeiten, eine bessere Messgenauigkeit zu erzielen:

- Der Teststreifen und der Sensor von *Bayer* haben Kontakte, die eine Blutmengenmessung zulassen. Diese könnte man sehr gut dementsprechend nutzen und in das Programm implementieren.
- Außerdem wäre über eine andere Art der Erkennung, dass sich Blut auf dem Kapillarfeld befindet, nachzudenken. Zwar wird die Messung innerhalb weniger Millisekunden auch im bereits vorhandenen Programmcode gestartet, jedoch ist es möglich, dass die Startwerte der Messung nicht erfasst werden und so Ungenauigkeiten entstehen.

7 Fazit und Schlusswort

Das entwickelte und in dieser Arbeit vorgestellte Blutglukosemessgerät mit integriertem Funkmodul zur Fernkontrolle in AAL stellt die grundlegenden Funktionen, die für ein solches telemedizinisches Gerät nötig sind, bereit. Die Steuerung über den Touchscreen erlaubt eine einfache und intuitive Bedienung, ein großer Vorteil gegenüber anderen Geräten, die durch viele Knöpfe und teils unübersichtliche Menüführung leicht verwirren.

Die Blutzuckermessung wird zum einen dadurch vereinfacht, dass der Bediener durch die einzelnen Menüpunkte geführt wird und außerdem kann durch die integrierte Funkübertragung des Messwertes eine Fernkontrolle durch den behandelnden Arzt durchgeführt werden. Dies erspart dem Patienten und dem Arzt viel Zeit und Kosten und ermöglicht trotzdem, ganz nach dem Grundsatz von *Ambient Assisted Living,* eine medizinische Betreuung.

In der Bearbeitung des Projektes wurde festgestellt, dass es sich bei der Messung des Blutzuckers um ein äußerst komplexes Feld handelt, das viele medizinische und chemische Kenntnisse erfordert. Deshalb ist der vorliegende Prototyp aufgrund der recht kurzen Zeit in Anbetracht der Komplexität dieses Themas noch fehlerbehaftet und muss überarbeitet werden.

Aufbauend auf dieser Projektarbeit wird eine grafische Darstellung auf dem PC mittels Java und OSGi im Rahmen einer Bachelorthesis erstellt, deren Aufgabe es sein wird, die über Funk übertragenen Blutzuckerwerte der Patienten in einer Datenbank abzuspeichern. Dies soll den Arzt in die Lage versetzen, den Verlauf der Messungen zu verfolgen und einen guten Überblick zu erhalten.

8 Abbildungsverzeichnis

9 Literatur- und Quellenverzeichnis

[1] J. N. Nestroy, Die Anverwandten, 1848.

[2] „Wikipedia,“ [Online]. Available: http://de.wikipedia.org/wiki/Telemedizin. [Zugriff am 4 10 2011].

[3] [Online]. Available: http://www.onmeda.de/arztbesuch/laborwerte/blut/blutzuckermessung-warum-blutzucker-messen--4443-2.html. [Zugriff am 30 10 2011].

[4] [Online]. Available: http://winfwiki.wi-fom.de/index.php/Resistive_vs._Kapazitive_Touchscreens. [Zugriff am 29 10 2011].

[5] [Online]. Available: http://de.wikipedia.org/wiki/IPv6. [Zugriff am 29 10 2011].

[6] [Online]. Available: http://www.elektronik-kompendium.de/sites/net/0812201.htm. [Zugriff am 29 10 2011].

[7] [Online]. Available: http://de.wikipedia.org/wiki/Serial_Peripheral_Interface. [Zugriff am 29 10 2011].

[8] [Online]. Available: http://www.erkenntnishorizont.de/robotik/uc/ucprogramm.c.php. [Zugriff am 29 10 2011].

[9] [Online]. Available: http://de.wikipedia.org/wiki/Touchscreen. [Zugriff am 29 10 2011].

[10] [Online]. Available: http://de.wikipedia.org/wiki/Blutzucker. [Zugriff am 30 10 2011].

[11] [Online]. Available: http://www.onmeda.de/arztbesuch/laborwerte/blut/blutzuckermessung-blutzucker-selbst-messen%3A-tipps-4443-6.html. [Zugriff am 30 10 2011].

[12] [Online]. Available: http://www.netdoktor.de/Diagnostik+Behandlungen/Laborwerte/Blutzuckermessung-und-Blutzucke-670.html. [Zugriff am 30 10 2011].

[13] [Online]. Available: http://de.wikipedia.org/wiki/Blutzucker. [Zugriff am 30 10 2011].

[14] [Online]. Available: http://de.wikipedia.org/wiki/Blutzuckerbestimmungsmethode. [Zugriff am 30 10 2011].

[15] [Online]. Available: http://www.mikrocontroller.net/articles/UART. [Zugriff am 10 11 2011].

[16] [Online]. Available: http://www.mikrocontroller.net/articles/AVR-Tutorial:_UART. [Zugriff am 10 11 2011].

[17] [Online]. Available: http://www.lcd-module.de/deu/pdf/grafik/dogs102-6.pdf. [Zugriff am 26 12 2011].

[18] [Online]. Available: http://www.contiki-os.org/p/about-contiki.html. [Zugriff am 24 1 2012].

[19] B. Esders, „Entwicklung, Implementierung und Evaluierung einer speichereffizienten, drahtlosen Kommunikation von eingebetteten Systemen mithilfe des IP-Protokolls basierend auf einem Multitasking-Betriebssystem für 8-Bit AVR Mikrocontroller.“ FHD, 2010.

[20] [Online]. Available: http://senstools.gforge.inria.fr/doku.php?id=contiki:examples. [Zugriff am 4 2 2012].

[21] [Online]. Available: http://www.sics.se/~bg/telos/html/a00208.html. [Zugriff am 4 2 2012].

[22] [Online]. Available: http://www.atmel.com/Images/doc8091.pdf. [Zugriff am 14 12 2011].

[23] [Online]. Available: http://www.atmel.com/Images/doc2585.pdf. [Zugriff am 11 12 2011].

[24] [Online]. Available: http://www.atmel.com/Images/doc2466.pdf. [Zugriff am 1 12 2011].

[25] [Online]. Available: http://www.atmel.com/Images/8272s.pdf. [Zugriff am 1 12 2011].

[26] [Online]. Available: http://www.mikrocontroller.net/articles/Operationsverst%C3%A4rker-Grundschaltungen#Der_Subtrahierer. [Zugriff am 22 11 2011].

[27] [Online]. Available: http://www.atmel.com/Images/doc8456.pdf. [Zugriff am 22 1 2012].

10 Eigenständigkeitserklärung

Hiermit versichere ich, dass ich das vorliegende Praxisprojekt selbständig verfasst habe.
Ich versichere, dass ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahren gewesen ist.

_____

Tanja Kleiner